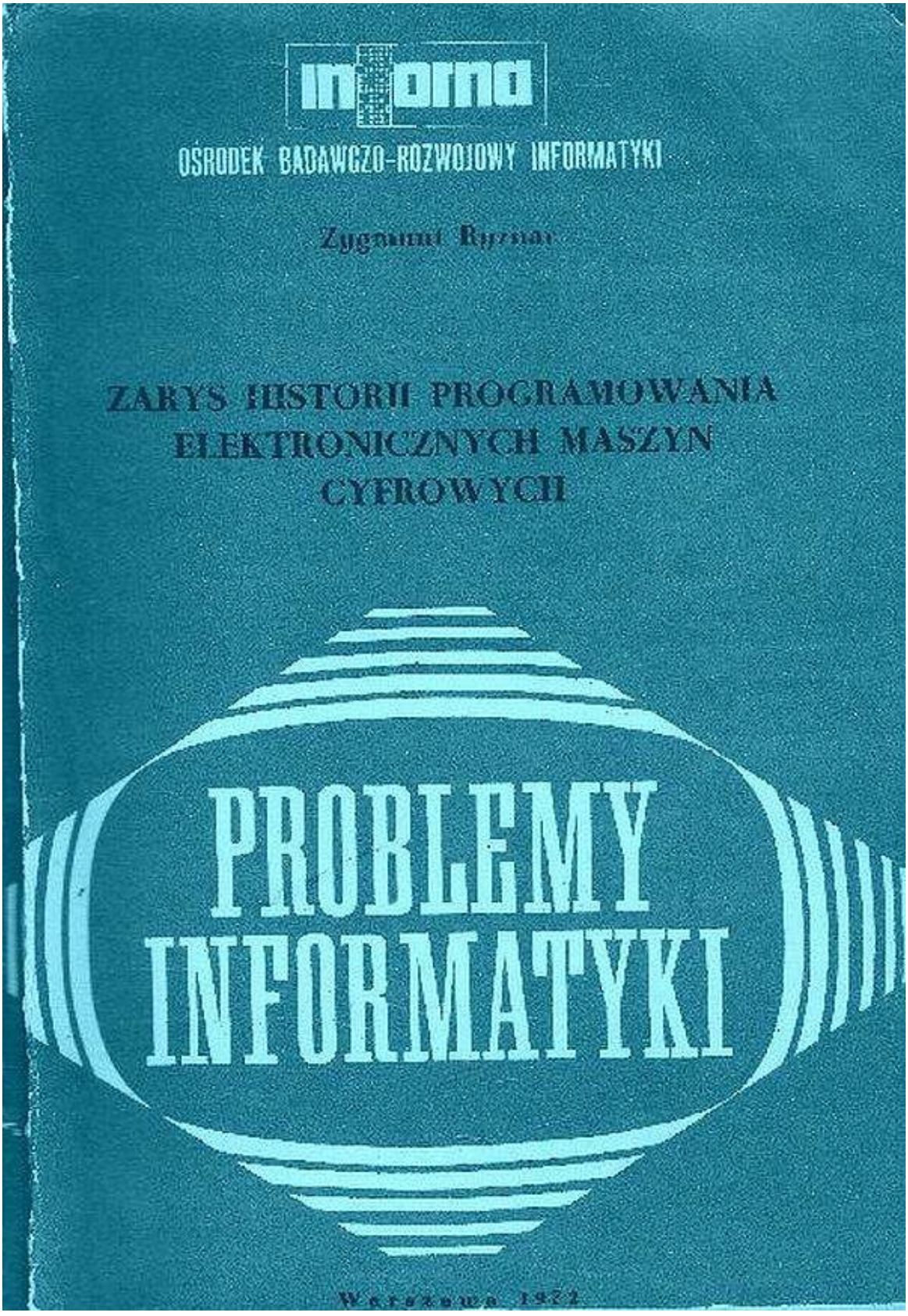


OKŁADKA PIERWOTNEGO WYDANIA W 1972 ROKU

Zygmunt Ryznar

Zarys historii programowania elektronicznych maszyn cyfrowych

(na tle rozwoju ich konstrukcji i zastosowań)

Wydanie pierwotne: Ośrodek Badawczo-Rozwojowy Informatyki •Warszawa 1972
Wydanie obecne: cyfrowe 2011 -v.final-2marzec

SPIS TREŚCI

I. Przesłanki i pierwsze koncepcje automatycznego liczenia
Pomysły Ch. Babbage'a • algebra Boole'a • zapis beznawiasowy J. Łukasiewicza • H. Aiken kontynuatorem idei Babbage'a • maszyna Turinga • zasługi Johna v. Neumanna • idea modyfikacji adresów.
II. Pierwsze koncepcje języków wyższego rzędu
Pojęcia interpretacji i kompilacji • język interpretacyjny Laninga i Zierlera • zasługi Rutishausera • idee Hopper i pierwsze kompilatory • metoda operatorowa Liapunowa • notacje N.Chomsky'ego i J.Backusa • pojęcia składni, semantyki i pragmatyki • notacja K. E. Iversona • zapis wiedeński • metoda Floyda • pierwsza propozycja schematów blokowych.
III. Przyczyny prac nad automatyzacją programowania oraz trudności wdrażania języków programowania
Pojęcie autokodu • efektywność poziomów programowania • struktura zastosowań poziomów • kryteria porównawcze języków • zalety języków wyższego rzędu • trudności wdrażania FORTRANu i COBOLu • statystyka języków programowania.
IV. Klasyfikacja języków programowania
Kryteria klasyfikacji • przykłady systemów interpretacyjnych • cechy języka symbolicznego • porównanie języków do obliczeń numerycznych z COBOLem • porównanie COBOLu z PL/I • cechy języka PL/I • języki wąskospecjalizowane • generatory programów • systemy operacyjne: pojęcie i funkcje, system operacyjny IBM 360, Honeywell OS/200, zestawienie porównawcze systemów operacyjnych.
V. Geneza języków powszechnego zastosowania
Geneza ALGOLu • języki algolopodobne • geneza FORTRANU • wpływ FORTRANu na inne języki • geneza COBOLu • prace organizacyjne, komitety • wersje • ANSI • zestawienie pierwszych kompilatorów języka COBOL • zestawienie pierwszych języków do przetwarzania danych • zapożyczenia COBOLu • język IDS • DIBOL • Compact COBOL, Rapidwrite • geneza języka PL/I.
VI. Zamienność języków i programów
Koncepcja uniwersalnego języka pośredniego • emulacja-symulacja • translacja • typy tłumaczeń • zamienność programów COBOLu, FORTRANU, ALGOLu.
VII. Rozwój programowania a rozwój konstrukcji maszyn
Rozwój urządzeń pamięciowych • mikroprogramowanie adresowość • definicja komputera • generacje maszyn • systemy wielomaszynowe •

geneza systemów wielodostępnych pracujących w podziale czasu.

VIII. Rozwój programowania a zastosowania elektronicznych maszyn cyfrowych

Uwagi ogólne o przeszłości i przyszłości: pierwsza zastosowania i perspektywy • statystyka i klasyfikacja zastosowań oraz powiązania z ilością języków programowania • generacje zastosowań • systemy zintegrowane.

IX. Języki programowania opracowane w Polsce

Oprogramowanie maszyny ZAM-41 • język EOL • translatory ALGOLu • MOST • LOGOL • FALA 68

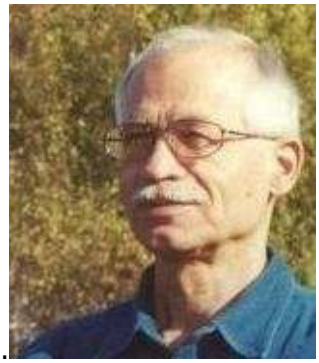
X. Ocena stanu dotychczasowego i tendencje rozwojowe programowania

Dorobek dotychczasowy • potrzeba metajęzyków i opartych o nie języków wąskospecjalizowanych • HELP • programowanie palindromiczne • COBOL CCF • języki tablicowe i tablice decyzyjne • systemy samoprogramujące.

Bibliografia

Materiały dodatkowe - nie występujące w wydaniu 1972 r.

1. **Kalendarium informatyki do 1971 roku**
2. **O tych co stworzyli podstawy informatyki i przemysłu komputerowego** • Herman Hollerith • Mauchley • Atanasoff • J. von Nemann • Howard Aiken • Konrad Zuse • Alan Turing • Grace Hopper • Kenneth H. Olsen • Steve Wozniak • Steve Jobs • Seymour CRAY • Edgar Frank Codd • Linus Benedict Torvalds • Wiliam (Bill) H.Gates
3. **Krótka historia przemysłu komputerowego**


Tyle czasu minęło...


Przedmowa autora do wydania cyfrowego 2011

Książka ukazała się prawie 40 lat temu w niskonakładowym wydawnictwie Ośrodka Badawczo-Rozwojowego Informatyki (OBRI) w Warszawie. Przypomniałem sobie o jej istnieniu podczas dyskusji na liście dyskusyjnej Sekcji Historycznej Polskiego Towarzystwa Informatycznego (PTI "Klio"). Umieściłem niektóre strony na forum i okazało się, że praca wzbudziła spore zainteresowanie. Potem dzięki Pani Marii Miller została graficznie zeskanowana w Bibliotece Cyfrowej Politechniki Warszawskiej i wówczas zdobyła duże uznanie. Wśród entuzjastów był Pan Włodzimierz Wypych, z którego inicjatywy i którego redaktorskim wysiłkiem praca została tekstowo zeskanowana, zweryfikowana i w dużej mierze zaprojektowana, tworząc podstawy niniejszej edycji cyfrowej. Korzystając z okazji składam mu podziękowanie za tak znaczny wkład. Dziękuję również Tym, którzy nadesłali uwagi techniczne i merytoryczne. Dzięki nim praca zyskała na jakości. Na etapie końcowej redakcji dokonałem znacznych zmian w tresci i wyglądzie, więc za ew. niedostatki obecnej wersji jestem odpowiedzialny wyłącznie ja.

Praca została napisana w 1971 roku (wyszła w 1972 roku). Wtedy były inne czasy. Nie było komputerów osobistych i internetu. Książkę trzeba było napisać na maszynie i przysłać tekst ostateczny do wydawcy czasem nawet już rok przed wydaniem. W maszynopisie możliwe były jedynie literowe poprawki i zmiana układu tekstu nie wchodziła w grę.

W sytuacji braku internetu upowszechnianie oraz zbieranie informacji było bardzo utrudnione. Z powodu ówczesnych ograniczeń dewizowych w bibliotekach brakowało literatury zagranicznej i zdobywało się ją głównie poprzez zakupy prywatne. Stąd pewne braki merytoryczne w opracowaniu, np. nie ma informacji o języku BCPL (Basic Combined Programming Language), z którego wywodzi się język C. O składni języka BCPL opracowywanego od 1966 roku przez Martina Richardsa z Univ. Cambridge usłyszano tak na prawdę dopiero w 1969 r na Spring Joint Computer Conference.

Do tekstu pierwotnego pracy nie wprowadzałem uzupełnień, poza niezbędnymi przypisami wynikającymi z upływu czasu.

Internetowych wersji edycji cyfrowej było kilka. Niniejsze wydanie zostało znacznie wzbogacone, gdyż na końcu zamieściłem kilka dodatków z historii informatyki.

Uwagi na temat opracowania proszę przesyłac na adres mailowy zygmunt.ryznar@gmail.com. Tych, którzy są szerzej zainteresowani moją osobą, zapraszam na Stronę domową: http://home.autocom.pl/zryznar/

Zygmunt Ryznar

Zasady przyjęte przy wydaniu cyfrowym

Celem naszym było uwolnienie opracowania od ograniczeń, jakie narzucała jej dotychczasowa forma wydawnicza w postaci powielonego maszynopisu. Chodziło o wydobycie tekstu i jego uczytelnienie przynajmniej w takim stopniu, aby mógł on stanowić podstawę dla dalej idącego opracowania, a ewentualnie także przygotowania nowego, rozszerzonego wydania. Uznając tę pracę za doniosły dokument polskiego piśmiennictwa informatycznego, starano się przy jego opracowaniu przestrzegać przyjętych ogólnie zasad edytorstwa w zakresie nowszych źródeł historycznych.

1. Wszystkie zmiany w tekście wskazano w przypisach, które, dla odróżnienia od przypisów oryginalnych, oznaczono datowaną sygnaturą. Bez zaznaczania poprawiono jedynie (nieliczne) błędy literowe oryginału.

2. Zachowano przyjętą przez autora konwencję notacji, na przykład sposób zapisu nazw języków programowania (ALGOL, ALGOLu, ALGOLem) czy danych bibliograficznych.

3. Dla uczytelnienia tekstu wyróżniono typograficznie słowa kluczowe języków formalnych, formuły logiczne i fragmentu kodu programów. Za pomocą kursywy wyróżniono wyrażenia potoczne wzięte z języków obcych.

4. Przypisy wprowadzono bezpośrednio po jednostce tekstu (akapicie lub tabeli), do której się odnoszą.

5. Zachowano paginację oryginału, ale bez eksponowania podziału na strony. Aby uniknąć dzielenia słów między stronami przyjęto, że nowa strona następuje bezpośrednio po słowie podzielonym w tekście drukowanym.

Od redaktora wspomagającego edycję wersji cyfrowej

<7>

WSTĘP

Praca niniejsza jest próbą zwięzłej syntezy dorobku w dziedzinie programowania elektronicznych maszyn cyfrowych do roku 1971 na tle postępu technicznego i rozwoju zastosowań. Nie jest ona podręcznikiem programowania ani też rozprawą teoretyczną. Przeznaczona jest w zasadzie dla praktyków (projektantów i programistów) jako lektura pogłębiająca ich zawodową wiedzę ogólną w zakresie elektronicznej techniki obliczeniowej (uwalniająca poniekąd od studiowania obszernej obcojęzycznej literatury specjalistycznej). Będzie przeciwdziałać zbyt wąskiej specjalizacji (polegającej na "przywiązaniu" do jednego typu maszyny, tej zainstalowanej w ośrodku obliczeniowym danego projektanta czy programisty) oraz stworzy pomost do dialogu pomiędzy teoretykami programowania i programistami-praktykami.

Ponadto wydaje się, że praca niniejsza służyć może projektantom jako zbiór wskazówek do wyboru języka programowania, odpowiedniego do określonego rodzaju zastosowań oraz do wyboru typu maszyny.

Podane informacje historyczne oparte są oczywiście na literaturze specjalistycznej, zaś próby analizy porównawczej języków podejmowałem na własne ryzyko.

Zdaje sobie sprawę z tego, że może nie udało mi się ująć wszystkich istotnych faktów historycznych dotyczących rozwoju programowania i uniknąć potknięć. Wiąże się to z trudnościami uchwycenia obfitego dorobku praktycznego i teoretycznego, ponadto rozrzuconego po wielu publikacjach często dla mnie niedostępnych [1] .

[1] *Dotyczy to m.in. publikacji na temat języka BCPL, komputera Colossus oraz Konrada Zuse (materiały o nim potem upowszechnił w internecie jego syn Horst Zuse). (Przyp.2010)*

<8>

Wreszcie wynika to ze zmienności, jakiej z upływem czasu ulegają języki i koncepcje teoretyczne. Ponadto skromna objętość opracowania nie pozwalała na omówienie szeregu spraw.

Odrębne zagadnienia stanowią trudności terminologiczne w sytuacji, kiedy każdy prawie język (i producent maszyn) operuje różnym aparatem pojęciowym, niekiedy jasno nie zdefiniowanym.

Pewną ilustracją kłopotów formalnych mogą być trudności w ustaleniu dat powstania poszczególnych języków. Różne źródła podają różne daty, często bez dostatecznego komentarza. Tymczasem nie bardzo wiadomo, co uważać za datę powstania języka: czy opracowanie pierwszej teoretycznej koncepcji (opisu formalnego) czy też wdrożenia translatora [1] .

[1] *Mam tutaj na myśli przede wszystkim kompilatory i interpretery. (Przyp.2010)*

Praca niniejsza nie powstała jako rezultat działalności instytucji (naukowej czy innej), jak również nie stanowiła przedmiotu konsultacji czy współpracy. Jest po prostu owocem kilkuletniego hobbystycznego wysiłku jednej osoby. Będę bardzo zobowiązany i wdzięczny za rzeczową krytykę, uwagi i uzupełnienia, które wzbogacą moją wiedzę zawodową oraz pozwolą na poprawienie materiałów w ewentualnym następnym wydaniu.

Proszę o nadsyłanie ich na adres: Kraków, ul. Radzikowskiego 66 m. 112 [1] .

[1] *Adres ten jest obecnie niektualny. Do korespondencji można wykorzystywać mój adres emailowy: zygmunt.ryznar@gmail.com (Przyp.2011)*

Na zakończenie niech mi będzie wolno złożyć podziękowania mojej żonie LUDMILE, bowiem praca niniejsza powstała kosztem naszego czasu rodzinnego.

<9>

I. Przesłanki i pierwsze koncepcje automatycznego liczenia

Jedną z pierwszych maszyn cyfrowych, w których operacje wykonywane były za pomocą układów elektronicznych, był ENIAC (*Electronic Numerical Integrator And Computer*) zbudowany w Stanach Zjednoczonych pod koniec II wojny światowej [1] .

[1] *Wiadomo, iż operacje ENIACa sterowane były kablowo (ale realizowane w elektronicznej technice lampowej) poprzez tzw. plugboards,wiredboards czyli tablice z otworami na wiele przewodów, przejęte prawdodopobnie z techniki programowania maszyn licząco-analitycznych systemu kart dziurkowanych, które w owym czasie były bardzo rozpowszechnione również na uczelniach amerykańskich. Pod względem obsługi programowej nowocześniejszy był mechaniczno-elektryczny kalkulator Z1 – zbudowany w latach 1936–1938 przez Konrada Zuse – pobierający rozkazy z taśmy dziurkowanej. Przez wiele lat ENIAC uważany był za pierwszy elektroniczny komputer na świecie, co mijało się z prawdą, gdyż na to miano zasługiwały też komputer ABC (Atanasoff-Berry Computer) zbudowany w Iowa Univ. w latach 1937-42 (arytmetyka dwójkowa, lampy próżniowe, kondensatorowe pamięci) i Colossus, zbudowany przy udziale Alana Turinga elektroniczny lampowy komputer deszyfrujący (do łamania kodu Enigmy). Na "prestiż" ENIACa działa pozytywnie fakt, iż z tych komputerów był najdłuzej eksploatowany (aż do 1955 roku). Pod względem patentowym pierwszeństwo zostało w 1973 roku przyznane – po wieloletnim procesie sądowym – komputerowi ABC. (Przyp.2010)*

Dopiero jednak kilka lat później powstały elektroniczne maszyny liczące o nowoczesnej organizacji działania: przechowujące program w pamięci, liczące w dwójkowym systemie, etc. [1]

[1] *Wcześniej – bo w 1941 roku – powstała maszyna licząca Z3 (Konrada Zusego). Liczyła binarnie, była programowalna (chociaż bez instrukcji skoku warunkowego), lecz nie przechowywała programu (miała pamięć przekaźnikową o wielkosci 64 słów) – tak samo potem było w przypadku komputera lampowego ENIAC. Obie maszyny nie spełniały więc wymagań zdefiniowanych przez Johna von Neumanna.*

Podstawą budowy tych maszyn były nie tylko postępy elektroniki i mechaniki, lecz również logiki i teorii automatycznego liczenia, pochodzące niekiedy sprzed kilkuset lat. Maszyny liczące od dawna stanowiły obiekt praktycznych koncepcji oraz filozoficznych spekulacji. Pierwszymi konstruktorami mechanizmów liczących byli słynni filozofowie i uczeni: Pascal, Leibniz, Łomonosow, Czebyszew. Niezależnie od praktycznych osiągnięć (sumator) Pascal w Myślach wypowiada sąd o maszynach arytmetycznych, porównując ich działanie do myślenia żywych istot.

Prawdopodobnie pierwszym autorem częściowo zautomatyzowanej maszyny był Müller,

który w 1786 roku przedłożył projekt maszyny do obliczania algebraicznych funkcji różnicowych. Maszyny tej przypuszczalnie nie zbudowano.

Uważa się, że ojcem rachunku automatycznego był natomiast Charles BABBAGE (1792-1871), angielski matematyk kierujący w latach 1828-1839 katedrą matematyki w Cambridge.

Zasługi Babbage'a dla rozwoju maszyn liczących były ogromne.

<10>

Dlatego też poświęcimy im sporo miejsca. Uczony ten 60 lat swojego życia poświęcił na opracowanie koncepcji i budowę fenomenalnych jak na XIX wiek mechanicznych automatycznych maszyn liczących. Niestety nie dane mu było sięgnąć celu. Pierwsza maszyna licząca na miarę jego koncepcji zbudowana została dopiero pod koniec lat czterdziestych naszego stulecia (1949 – EDSAC). Zbudowano ją z elementów elektronicznych nie istniejących w czasach Babbage'a. Projekt angielskiego uczonego zawierał główne koncepcje współczesnego komputera: pamięć, urządzenie arytmetyczne wejście z maszynowego nośnika informacji (z kart dziurkowanych), wyjście poprzez urządzenie piszące itp.

Idee Babbage'a ujmowane są następująco. [C1]

a) maszyna licząca powinna wykonywać wszystkie operacje arytmetyczne,
b) program wprowadza się na kartach dziurkowanych,
c) maszyna powinna przechowywać rezultaty w celu późniejszego ich wykorzystania,
d) należy zapewnić możność wyboru pomiędzy działaniami w zależności od wyniku obliczeń.

Babbage stale pracował nad doskonaleniem swych pomysłów. Po 10 latach pracy nad projektem "maszyny różnicowej" w 1823 roku przy dotacji rządowej rozpoczął jej budowę. Po kolejnych 10 latach budowę przerwano z różnorakich powodów (zatarg z inżynierem prowadzącym budowę, trudności finansowe, kłopoty technologiczne, a przede wszystkim pomysł nowej doskonalszej maszyny). Nowa maszyna zwana analityczną (*analitical engine*) miała wykonywać dowolne obliczenia oraz pracować według innych idei Babbage'a.

<11>

Pamięć ("magazyn") tej maszyny składać się miała z 1000 rejestrów po 50 kół cyfrowych w każdym. W zależności od rozkazu każde koło mogło się łączyć z arytmometrem ("fabryką") lub innymi częściami maszyny. Jako ciekawostkę można podać to, że Babbage za największe swoje osiągnięcie uważał nie koncepcję maszyny analitycznej, lecz opracowanie algebry opisującej ruchy poszczególnych części maszyny. Można ten wysiłek uważać za próbę sformułowania teorii działania maszyn liczących.

W tym samym czasie żył również w Anglii jeden z najwybitniejszych logików XIX wieku George Boole (1815-1864). Stworzył on algebrę logiki, znajdującą szerokie zastosowanie w teorii elektronicznych maszyn cyfrowych. Właśnie od Boole'a pochodzi znakowanie operacji alternatywy ("dodawania" logicznego) i koniunkcji ("mnożenia"

logicznego). Kontynuatorami algebry Boole'a byli m.in. de Morgan, Porecki i Schröder oraz Jevons. Ten ostatni, oprócz rozpraw teoretycznych posiada w dorobku pierwszą (1869) maszynę do rozwiązywania zadań logicznych. Zastosowanie algebry Boole'a do opisu działań obwodów przełączających zaproponował w 1938 roku Claude Shannon (jak wiadomo, obwody równoległe przedstawiają operację "lub" zaś szeregowe – operację "i").

Duże znaczenie dla rozwoju automatyzacji programowania posiadają prace wybitnego polskiego logika Jana Łukasiewicza (1878-1956), który w 1910 roku pracą "Zasada sprzeczności u Arystotelesa" "" zapoczątkował logikę matematyczną w Polsce. Uczony ten stworzył w Warszawie ośrodek logistyczny o światowej sławie. Jego koncepcja beznawiasowego zapisu wzorów algebraicznych wykorzystana została do translacji wielu języków programowania. Znana jest w świecie jako tzw. *Polish Notation* (polski zapis). Rozróżnia się zapis prosty (*Direct Polish Notation*) oraz

zapis odwrotny (*Reverse Polish Notation*). Właśnie ten ostatni używany jest z reguły w technice kompilacji.

Wyrażenie: `A x B – C x D` w polskiej notacji w zapisie prostym wygląda `– x A B x C D`. W zapisie prostym znaki działań podawane są przed każda parą argumentów, zaś w zapisie odwrotnym – po parze (`A B x C D x –`). Od strony technicznej zapis Łukasiewicza realizowany jest za pomocą tzw. pamięci stosowej (*push-down memory*), przedstawiającej jak gdyby "stos" rejestrów, z których tylko jeden (górny) może kontaktować się z otoczeniem. Każde wejście-wyjście powoduje więc przemieszczenie się danych w rejestrach.

Po wielu latach zapomnienia idee Babbage'a odżyły. W 1930 roku dr Howard Alken z uniwersytetu w Harvard opisał model automatycznej maszyny liczącej opartej o XIX-wieczne koncepcje angielskiego matematyka.

Opis ten wykorzystano przy budowie maszyny przekaźnikowej MARK I (1937-1944).

Również w latach 30-tych A.M. Turing podał koncepcję abstrakcyjnej maszyny liczącej nazwanej później "maszyną Turinga". W odczycie wygłoszonym w 1936 roku w Londyńskim Towarzystwie Matematycznym przedstawił on teoretyczny model uniwersalnej abstrakcyjnej maszyny liczącej [1]. Maszyna ta wyposażona była w nieskończenie (z obu stron) długą taśmę z zapisanymi symbolami. Wzdłuż taśmy przesuwała się (!) głowica czytająco-pisząca sprzężona z układem sterującym. Praca maszyny polegała na odczytaniu symbolu z klatki na taśmie, a następnie na porównaniu go z dotychczasowym "swoim" stanem. W przypadku różnic następowało przejście do nowego stanu i obliczenie nowego symbolu, który był zapisywany na taśmie w miejsce symbolu poprzedniego.

[1] *Dalsze prace nad koncepcją "maszyny Turinga" zostały przerwane przez II wojnę światową. Po wojnie, pracując w National Physical Laboratory, Turing uczestniczył w budowie maszyny lampowej ACE (Automatic Computing Engine). A.Turing był również szefem zespołu, który zbudował komputer Colossus. Posiada on również pewne zasługi w pionierskim zastosowaniu koncepcji biblioteki podprogramów, ważnego etapu w rozwoju programowania.*

<13>

Przez usunięcie założenia o nieskończoności taśmy stworzono pojęcie automatu skończonego. Do automatu skończonego wprowadza się jakiś tekst i po odpowiednim przetworzeniu otrzymujemy nowy ciąg symboli. Wykorzystując zasadę "czarnej skrzynki", dokonuje się np. modelowania maszyn matematycznych na innej maszynie.

Największe zasługi dla rozwoju teorii działania nowoczesnych maszyn posiada John von NEUMANN (1903-1957). Ten amerykański matematyk węgierskiego pochodzenia już w 1936 roku, niezależnie od Francuza Couffignala, zaproponował zastosowanie dwójkowego systemu liczenia (zamiast dziesiętnego) w maszynach liczących. Będąc konsultantem w Moore School of Electrical Engineering w Filadelfii, pracował nad ulepszeniem maszyny ENIAC, zbudowanej przez Eckerta i Mauchly'ego. Jak wiadomo, ENIAC posiadał bardzo małą pamięć (dla 20 liczb) i dlatego nie mógł przechowywać programu. Konstruktorzy maszyny zdawali sobie sprawę z tej podstawowej wady i zaproponowali użycie dla następnych maszyn pamięci na ultradźwiękowych liniach opóźniających.

Niemniej jednak wszechstronne badania nad optymalnymi charakterystykami maszyn zostały przeprowadzone dopiero przez Neumanna i jego współpracowników (Goldstine, Burks). Badania te wykazały, że pojemność pamięci powinna wynosić (w owym czasie) co najmniej 4096 słów (o długości 10-12 znaków dziesiętnych), że powinna ona przechowywać zarówno dane jaki program oraz że może wyniknąć potrzeba zastosowania pamięci pomocniczej.

Grupa Neumanna stworzyła podstawy projektowe wielu maszyn (EDVAC,

<14>

SEAC, EDSAC i inne) John von Neumann był wszechstronnym uczonym i wniósł duży wkład w rozwój takich dziedzin jak mechanika kwantowa, logika matematyczna i teoria gier. W serii tzw. raportów publikowanych na przestrzeni lat 1945-1947 podał on fundamentalne zasady budowy maszyn matematycznych:

a) zarówno dane jaki program mogą być wyrażone w systemie binarnym,

b) program powinien być przechowywany w pamięci,

c) ponieważ rozkazy są kodowane i przechowywane tak samo jak liczby, można je przetwarzać arytmetycznie w celu otrzymania nowych rozkazów.

Niezależnie od Neumanna, w 1949 roku Wilkes wskazał, że należy dokonywać operacji arytmetycznych na adresach, osiągając przez to znaczne skrócenie wielkości programu (Jak wiadomo, modyfikacja adresów pozwala na tworzenie tzw. pętli w programie). Jak więc widzimy, nie od razu odkrywano oczywiste dzisiaj dla nas zasady działania komputerów. Była to droga ewolucji, którą kiedyś zapoczątkował Charles Babbage a przyspieszył Alan Turing, Howard Alken i John von Neumann.

<15>

II. Pierwsze koncepcje języków wyższego rzędu

Wraz z rozwojem urządzeń pamięciowych i pilną potrzebą ułatwienia programowania pojawiły się koncepcje ujęcia czynności sterujących (operacyjnych) oraz obliczeniowych i logicznych, nie tylko w postaci układów konstrukcyjnych (tj. mikroprogramów) i kodów maszynowych, lecz również za pomocą instrukcji specjalnych wymagających uprzedniego tłumaczenia.

W zależności od sposobu dokonywania tłumaczenia takich instrukcji rozróżniamy systemy interpretacyjne i kompilacyjne. Systemy interpretacyjne stanowią rozwinięcie idei podprogramów. Po rozpoznaniu każdej instrukcji wywoływany jest odpowiedni podprogram i następuje jego wykonanie, po czym pobierana jest następna instrukcja. W systemach kompilacyjnych cały program źródłowy (*source program*) jest najpierw tłumaczony na język wewnętrzny (*object program*), a dopiero później wykonywane są obliczenia w zakresie całego programu.

Pierwsze interpretacyjne programy zastosowano w celu wykonywania operacji w zmiennym przecinku na maszynach pracujących w stałym przecinku oraz symulacji pracy maszyny wieloadresowej na maszynie jednoadresowej. Jednym z pierwszych języków interpretacyjnych był interpretacyjny system kodowania wyrażeń algebraicznych opracowany w latach 1952-1953 przez J.H. Laninga i N. Zierlera w MIT dla maszyny Whirlwind.

Podobne prace dla systemów kompilacyjnych prowadził w Szwajcarii Heinz Rutishauser zatrudniony w Federalnym Instytucie Technologicznym w Zürichu. Już w 1952 roku przedstawił

<16>

on metody umożliwiające wprowadzenie do maszyny wyrażeń algebraicznych w zwyczajnej formie. Ponadto Rutishauser zaproponował bardzo prostą i zwięzłą instrukcję do organizacji pętli `for k = (1) 10`.

Ograniczoną próbą zastosowania matematycznej notacji na wejściu był system SHORT CODE zaproponowany w 1949 roku przez J. Mauchley'ego (współtwórcy ENIACa) a zastosowany w 1952 roku dla maszyn UNIVAC i BINAC [1].

[1] *Warto odnotować informację, że już w 1945 roku Konrad Zuse opracował język programowania wysokiego poziomu, zwany Plankalkül, który używany był m.in. do modelowania ruchów szachowych.*

W 1953 roku dla maszyny IBM 701 opracowano Speed Coding System, w opracowaniu którego uczestniczył J. Backus. W 1956 roku pojawiają się pierwsze publikacje na temat MANCHESTER UNIVERSITY AUTOCODE. Firma GEC Computers na bazie tego języka opracowała w 1962 roku "autokod" (*autocode*) dla maszyny FERRANTI MERCURY.

Termin "kompilator" został wprowadzony przez Dr Grace HOPPER, zatrudnioną w firmie Remington Rand jako szef programowania. Zaproponowała ona użycie języka

angielskiego do pisania instrukcji.

Już w 1952 roku opisała działanie kompilatora a następnie kierowała a pracami, w wyniku których powstały zapewne pierwsze w świecie kompilatory AO i A1, w 1955 roku powstaje wersja A2, a następnie AT3 (nazwana ARITH-MATIC). Pierwszy opis tego języka znany był w 1957 roku. W tym samym roku pojawił się opis języka FLOW-MATIC, ukierunkowanego na zastosowanie ekonomiczno-administracyjne.

FLOW-MATIC, zwany początkowo jako język B-0, charakteryzował się szerokim użyciem języka angielskiego zarówno do pisania instrukcji jak i do opisu danych. Język ten stanowił

<17>

później jedną z podstaw języka COBOL (1959). Na poparcie tego twierdzenia przytoczymy parę instrukcji języka FLOW-MATIC, prawie "żywcem" wziętych do COBOLu:

```
ADD A B C
DIVIDE nd_1 BY nd_2 GIVING nd_3 [1]
MOVE nd_1 TO nd_2,
READ ITEM n_d IF END OF DATA GO TO OPERATION ...
```

[1] *Skróty* `nd, nd_1, ...` *wprowadzone dla zwięzłości w przykładach, oznaczają "nazwa danej", "nazwa danej 1", ... (Przyp.2010)*

Ważnym problemem programowania jest algorytmizacja zadania, tj. jednoznaczne określenie reguł działania procesu obliczeniowego, uwzględniające wszystkie możliwe kombinacje i etapy. Przy okazji warto zaznaczyć, że słowo "algorytm" pochodzi według wg. [T3] od nazwiska żyjącego w IX wieku uzbeckiego matematyka AL-HOREZMI [1] .

[1] *W innych przekazach niż radzieckie jest on uczonym arabskim, znanym jako Muhammed ibn Musa al-Chorezmi (czyli "Muhammed syn Musy z Chorezmu"), który ok. 820 roku opisał pozycyjny system liczbowy stosowany w Indiach i sposoby liczenia w tym systemie. (Przyp.2010)*

Na lata 1952-1953 przypadają początki opracowania w ZSRR (w Matematycznym Instytucie AN ZSRR) pod kierownictwem A. A. Liapunowa – tzw. operatorowego opisu algorytmów. U podstaw metody leży założenie, że algorytm składa się z szeregu powtarzalnych arytmetycznych i logicznych etapów. Każdy z nich jest realizowany za pomocą tzw. operatora, reprezentującego grupę elementarnych działań, wykonanie których może przebiegać automatycznie w powiązaniu z tzw. programującym programem. Program zewnętrzny (pisany przez programistę) zawiera takie informacje jak:

a) typ operatora (A – arytmetyczny, P – logiczny, `F(i)` – przeadresowanie, `E(m;n)` – skok do operatorów z numerami `m,m-1, ... n`) itp. nr porządkowy operatora, kierunki skoków,

b) wzory matematyczne dla operatorów arytmetycznych,

parametry przeadresowania,

d) podprogramy itp.

Są to informacje o operatorach standardowych. Operatory niestandardowe

<18>

muszą być całkowicie zaprogramowane w języku wewnętrznym. Główne czynności sprowadzają się do wyboru metody obliczeń, sporządzenia logicznego schematu programu (przez wypisanie ciągu operatorów), określeniu wzorów i parametrów oraz zaprogramowania niestandardowych fragmentów programów.

Oto przykład logicznego schematu w metodzie operatorowej:

Programujący program opracowano m.in. dla maszyny Strieła-4

Opracowanie języka programowania wymaga zdefiniowania szeregu takich elementów, jaki alfabet (wykaz dozwolonych znaków), wyrażenie (instrukcja) i zdanie (zespół instrukcji zakończony znakiem końca, np. kropka), deklaracje (np. dotyczące opisu danych), dyrektywy operacyjne (związane z organizacją translacji), reguły tworzenia procedur (bloków, paragrafów), reguły wzywania podprogramów i umieszczania wstawek w innych językach itp.

Potrzebę ograniczonego słownika znaków sygnalizował już Turing w koncepcji swojej abstrakcyjnej maszyny. Ze znaków tych proponował tworzenie dowolnych kombinacji.

Problemy konstrukcji, interpretacji i stosowalności języka ujmowane są jako składnia, semantyka i pragmatyka. Składnia określa, jakie sekwencje znaków są dopuszczalne (np. w postaci instrukcji), przykładowo *A + B* może stanowić prawidłową konstrukcję w języku "XX", podczas gdy *+AB* jest nielegalne. Semantyka obejmuje reguły określania znaczenia dopuszczalnych kombinacji, zaś pragmatyka – reguły stosowalności zarówno semantyki

<19>

jak i składni.

Wszystkie trzy powyższe działy powinny być opisane w sposób jednoznaczny, a jest to możliwe dzięki zastosowaniu metod sformalizowanego zapisu. Tak się złożyło, że dotychczasowe badania dotyczyły głównie składni, zaś semantyka i pragmatyka czekają dopiero na opracowanie, a właściwie stworzenie. Badania nad składnią zapoczątkował w 1956 roku Chomsky.

Zdefiniował on podstawowe elementy pragmatyki formalnej,

a) podzbiór terminalny, tj. słownik zwrotów języka (małe litery),

b) słownik nazw strukturalnych, tj. zmiennych (duże litery),

c) produkcje tj. prawa podstawiania.

Odnośnie tego ostatniego elementu warto dodać, że pojęcie podstawienia, zdefiniowane

przez Posta i Thuego, stanowi kluczowy kanon w metodach zapisu składni.

Duże zasługi w rozwoju notacji posiada J.BACKUS. W notacji (1959) swojej zmienił on znak podstawienia N.Chomsky'ego z *–>* na *::* zaś wprowadzenie nawiasów *<*, *>* pozwoliło na zamianę dużych liter przez opisowe słowa lub frazy.

Notacje Backusa wykorzystano w ALGOL REPORT (1960) zredagowanym przez Naura. Stąd też pochodzi skrótowa nazwa notacji BNF (*Backus Normal Form* lub *Backus Naur Form*). Rozwinięciem zapisu BNF jest metoda van Wijngaardena, przewidująca nieskończoną liczbę produkcji. Niekiedy spotkać można oznaczenie PNF (*Paniani Backus*

Form). Paniani [znany też jako Panini] był nauczycielem perskim, żyjącym ok. 600 lat p.n.e., który opracował system notacji zbliżony do formy BNF [B4].

K. E. Iverson zaproponował (1964) notację w pewnym sensie pośrednią pomiędzy "rozwlekłym" zapisem J.Backusa i "anonimowym" zapisem N.Chomsky'ego. Notacja Iversona umożliwia precyzyjny opis złożonych procesów sekwencyjnych. Oto jej przykłady:

a) operacje arytmetyczne: $+ \quad - \quad \times \quad \div$

b) operacje logiczne:

$I \quad W \leftarrow U \wedge V$

$LUB \; W \leftarrow U \vee V$

c) inne:

przesuń na lewo X o K miejsc $Z \; -- \; K \uparrow X$

przesuń na prawo X o K miejsc $Z \; -- \; K \downarrow X$

W ośrodku IBM w Wiedniu opracowano (publ. 1968) tzw. wiedeńską metodę opisu języków programowania. Użyto jej m.in. do opisu języków PL/I i ALGOL 60. W metodzie tej zastosowano aparat formalny tzw. obiektów abstrakcyjnych. Do opisu struktury syntaktycznej oraz procesów używa się grafów (w postaci drzew). Każdy wierzchołek grafu procesu związany jest z instrukcją. Wykonanie instrukcji powoduje usunięcie związanego z nią wierzchołka drzewa sterowania.

Również o graficzną metodę przedstawiania programu opiera się metoda Floyda. Stosuje się w niej schematy blokowe, składające się z bloków operacyjnych i warunkowych, połączonych strzałkami. Dla każdej strzałki w sposób formalny (stosując specjalną notację) podaje się warunki dotyczące zmiennych. Program sprawdzany jest przez przypisanie zmiennym aktualnych wartości i porównanie ich z warunkami. Metoda ta stanowi więc pewną analogię do znanej metody sprawdzania programów przez tzw. suche przebiegi. Opis metody Floyda podany jest w publikacji A. Mazurkiewicza [M2].

Pierwszy system oznaczeń graficznych (schematów blokowych) został zaproponowany przez Burksa, Goldstine'a i von Neumanna, przy czym był on wygodny dla maszyn jednoadresowych, podobnych do tych. które zostały zaprojektowane w Institute of Advanced Study. Mimo, iż schematy te były ukierunkowane na konkretne maszyny, zapoczątkowały one metodę uniwersalną, pozwalając na dokładne wyrażenie logiki programów niezależnie od rodzaju maszyny. Jest to więc pewne przybliżenie do języka uniwersalnego.

III. Przyczyny prac nad automatyzacją programowania oraz trudności wdrażania języków programowania

Do dziś jeszcze spotykamy przeciwników posługiwania się autokodami. Przy okazji wyjaśniamy, że przez "autokod" rozumiemy tutaj język AUTOmatycznego KODowania, tj. taki język, który na podstawie jednej zewnętrznej instrukcji generuje sekwencję rozkazów w języku wewnętrznym. Tak więc definicja ta nie obejmuje języków symbolicznych o współczynniku kodowania "jeden do jeden", jako że stanowią one po prostu zmodyfikowaną wersję języka wewnętrznego (przykładem takiego języka jest w zasadzie SSK – *Sistiema Simboliczeskowo Kodirowanija* – dla emc Mińsk 32).

Wyraz "autokod" pochodzi prawdopodobnie od nazwy jednego z pierwszych języków MANCHESTER UNIVERSITY AUTOCODE, opracowanego w II połowie lat 50-tych w Wielkiej Brytanii. Często, naszym zdaniem, niesłusznie pojęcie autokodu utożsamiane jest wyłącznie z językami symbolicznymi, co zostało być może spowodowane nazwą IBM-owskiego języka symbolicznego AUTOCODER.

Podobnie, jak przy każdym innym rodzaju postępu, wprowadzanie

<22>

autokodów natrafiało na duże trudności, noszące przede wszystkim charakter psychologiczny.

Obecnie pojęcie "Autokod" traktowane jest w literaturze jako archaizm, użyty do nazwania pierwszych eksperymentalnych języków z FORTRANEM I włącznie, w których – dzięki zastosowaniu notacji zbliżonej do matematycznej wyeliminowano współczynnik kodowania 1:1 oraz pozostawiono translatorom sprawę alokacji pamięci.

Już w 1953 roku, a więc wtedy, kiedy przeprowadzono dopiero pierwsze eksperymenty nad automatyzacją programowania, na jednym z pierwszych posiedzeń ACM (Association for Computing Machinery) omawiano sprawę tzw. "prymitywistów" (zwolenników języka wewnętrznego) oraz "żołnierzy postępu" (*space cadets*). Ci ostatni posądzani byli o propagowanie luksusu programowania wg "śmiesznych" schematów ograniczających myślenie. Zarówno jedni jaki drudzy dysponowali pewnymi argumentami. Również i dzisiaj sprawa poziomu programowania nie jest zupełnie (przynajmniej dla niektórych) wolna od nieporozumień. Wydaje się jednak, że sama praktyka (popularność zewnętrznych języków programowania) spór ten rozstrzygnęła.

Najpoważniejszym argumentem przeciwko autokodom (a w szczególności przeciwko językom wyższego rzędu typu ALGOL, COBOL) jest mała efektywność programu wewnętrznego, jaki generują, oraz większe zapotrzebowanie na pamięć operacyjną. Argumenty te, aczkolwiek całkiem słuszne (jeśli poniekąd rozpatrywać je abstrakcyjnie), nie przekonywują, jeśli zważymy, te w ostatnim okresie nastąpił ogromny wzrost szybkości działania maszyny i powiększenie pojemności pamięci operacyjnej,

wspomaganej przez pamięci

<23>

pomocnicze o dostępie wyrywkowym. Opieranie się na porównaniach pracy małych maszyn (dla których autokody są rzeczywiście mało efektywne) jest nie do przyjęcia, zważywszy, że produkcja ich należy do przeszłości (mamy na myśli maszyny o szybkości kilkuset operacji czy kilku tysięcy operacji na sekundę oraz wyposażenie w 4 – 6K pamięci operacyjnej). Straty czasu tak znaczne przy kompilatorach lat 60-tych [X1] zostały znacznie (niekiedy kilkakrotnie) zmniejszone.

Niemniej jednak w pewnych sytuacjach wybór elementarnego języka programowania można uznać za słuszny. W szczególności dotyczy to programów standardowych czy też translatorów (niekoniecznie) oraz niektórych programów obliczeniowych o dużej częstotliwości przetwarzania (np. codziennie) i operujących na dużej ilości danych.

Porównując efektywność języków należy wziąć pod uwagę kwalifikacje programistów i jakość translatorów, jako elementy zmienne przy każdym porównaniu. Spotyka się opinie, że program przetłumaczony z języka wyższego rzędu (przez dobry kompilator) dorównuje klasą programowi w języku elementarnym napisanym przez programistę średniej klasy.

Szczególny opór, np. w stosunku do COBOLu, przejawiają programiści, którzy uprzednio długo programowali w języku symbolicznym lub wewnętrznym. Przestawienie się na inne sposoby programowania stanowi dla nich trudną barierę.

W poniższej tabeli przedstawimy zależność kosztu i efektywności od poziomu języka (wg. [B6]):

<24>

		I poziom język symb. 1:1	II poziom język symb. + makro instr.	III poziom typ COBOL ALGOL	IV poziom generatory
Dobry programista	efektywność programu w jęz. wewn.	95%	90%	60-85%	45-65%
	koszt (indeks)	100	90	40-60	20-40
Zły programista	efektywność	70%	75%	50-70%	40-60%
	koszt	110	100	50-70	30-50

Wg tego samego źródła, struktura zastosowań poszczególnych poziomów języków przedstawia się następująco:

Poziom języka	1962	przewid. 1972
1. Symboliczny 1:1	65%	25%
2. Symboliczny z makroinstr.	20%	15%
3. Kompilatory	5%	40%

4. Generatory	10%	20%

Inne późniejsze (1969) badania amerykańskie oraz niemieckie[B5] wykazały, że w zastosowaniach administracyjno-ekonomicznych (poza podprogramami standardowymi) język wyższego rzędu np. COBOL (lub inny do przetwarzania danych) dorównuje językom symbolicznym.

<25>

Przy porównywaniu języków programowania należy stosować nie jedno kryterium, lecz brać pod uwagę cały zespół czynników:

1. Czas przejścia od "problemu" do programu, czyli uczenie się języka oraz zdefiniowanie algorytmu odpowiednio do wymogów języka. Jest to czynnik, na który bardzo rzadko zwraca się uwagę.
2. Czas pisania programów.
3. Czas uruchomienia programów.
4. Efektywność translatora:czas translacji,pojemność pamięci operacyjnej niezbędnej dla translacji, efektywność programu w języku wewnętrznym (po translacji) którą można określić Jako ilość rozkazów wewnętrznych, pojemność pamięci koniecznej do przechowywania programu i danych oraz czas biegu programu (czyli czas przetwarzania).

Zalety języków wyższego rzędu ująć można następująco:

1. pracochłonność programowania jest co najmniej kilkakrotnie niższa,
2. języki te z reguły są łatwiejsze do opanowania (m.in. ze względu na użycie słów naturalnego języka),
3. łatwiej i krócej uruchamia się programy (krótszy program zewnętrzny, łatwiejsza diagnostyka błędów, mniej pomyłek programisty, łatwiejsze korekty itp.),
4. program stanowi sam w sobie niezłą dokumentację (odnosi się to szczególnie do COBOLu), co nie jest bez znaczenia dla organizacji podziału pracy, fluktuacji kadr, wprowadzania zmian do programu oraz wymiany doświadczeń.

<26>

5. względna niezależność programu (napisanego w języku powszechnego użycia: COBOL, FORTRAN, ALGOL) od typu maszyny, co umożliwia użytkownikowi zachowanie ciągłości przetwarzania przy zmianie typu maszyny (po dokonaniu pewnych przeróbek nie zmieniających jednak podstawowych rozwiązali programowych) zaś programiście – zachowanie poprzednich kwalifikacji,

6. logika złożonego programu może zostać przedstawiona stosunkowo prosto (syntetycznie), co ułatwia tzw. suche (przy biurku) sprawdzanie biegu programu,

7. programista może bardziej koncentrować się na treści problemu, dla którego należy opracować program, niż na technice programowania i właściwościach konstrukcyjnych maszyny; dlatego też możliwe jest posiadanie dobrych analityków, będących równocześnie dobrymi programistami w COBOLu,

8. języki wyższego rzędu są niezastąpione w systemach konwersacyjnych człowiek-maszyna typu pytanie-odpowiedź, gdzie chodzi o łatwy i szybki dostęp do maszyny dla nieprogramistów.

Niektóre z wyżej podanych zalet stają się bardzo ważne w sytuacji ostrego deficytu kadr specjalistów oraz postępu technicznego (zmiany maszyn, a więc i języków maszynowo zorientowanych). Ponadto języki wyższego rzędu są szczególnie przydatne w przypadku konieczności szybkiego oprogramowania systemu i przy tworzeniu tzw. pierwszych wersji programów (mających na celu sprawdzenie ich logiki) wyprzedzających prace długofalowe nad optymalnymi pakietowymi rozwiązaniami.

Jeśli mimo tych zalet, występowały duże trudności wdrażania języków wyższego rzędu, to wynikało to nie tyle z oporu użytkowników ile niechęci producentów. Można twierdzić, że właściwie żaden język nie uzyskiwał "prawa obywatelstwa" bez walki.

<27>

Oto przykłady:

Firmowy język IBM – FORTRAN opracowany w latach 1954-1955 (niewątpliwie jeden z najstarszych języków programowania wśród obecnie stosowanych) aż do 1961 roku był ignorowany przez innych producentów (1961 – pierwsza wersja FORTRANU I dla UNIVAC Solid State 80), zaś późniejsza akceptacja języka powodowana była głównie chęcią przejęcia klientów od IBM.

Z kolei firma IBM usiłowała zbojkotować język COBOL; mimo własnego udziału w pracach nad jego opracowaniem. Twierdząc, że język ten jest niedostatecznie zdefiniowany (co było częściowo prawdą), firma ta opracowała własną kontrpropozycję: języki COMMERCIAL TRANSLATOR i AUTOCODER. Zważywszy, że w owym czasie 80% ogółu użytkowników maszyn było klientami IBM, sprawa przybrała poważny obrót.

Nie tylko zresztą IBM występował przeciwko COBOLowi. Firma NCR pozostawała przy języku NEAT (*National Electronic Autocoding Techniques*), zaś Honeywell zaproponował inny język do przetwarzania danych FACT (*Fully Automatic Compiling Technique*), który pod względem swoich możliwości niekiedy przewyższał COBOL.

W pierwszym okresie istnienia COBOLu (1960) jedynie firmy RCA i Remington Rand opracowały translatory dla tego języka.

Chcąc pokonać opór producentów rząd USA ogłosił w 1964 roku blokadę zakupów (dla rządowych instytucji) maszyn bez translatorów języka COBOL i ogłosił politykę konkursowych zakupów maszyn wyposażonych w COBOL (chodziło o zakup priorytetowy maszyn z najlepszymi translatorami). Ponieważ zakupy rządowe były dosyć znaczne, np. tylko w 1963 roku US Air Force zakupił 500 maszyn, taktyka ta przyniosła pełny sukces.

Podwójną grę w stosunku do COBOLu początkowo prowadziła

<28>

firma HONEYWELL. Mimo, iż przedstawiciele tej firmy wchodzili w skład zespołów roboczych COBOLu, utrzymywała ona w tajemnicy praca prowadzone nad językiem FACT. Pierwszy opis tego języka opublikowany na jesieni 1959 roku był kompletnym zaskoczeniem dla Komitetu COBOLu. Pod wpływem presji rządu firma ta zaakceptowała jednak COBOL, a nawet rozpoczęła prace nad rozwojem tego języka (dochodząc do bardzo dobrych translatorów). Język FACT został wdrożony jedynie na maszynie H-800.

Mimo początkowych trudności wdrożeniowych stosunkowo szybko przystąpiono do opracowywania różnego rodzaju języków automatycznego programowania (czy też raczej należałoby użyć określenia – kodowania) i wkrótce doszło nawet do wyraźnej ich nadprodukcji.

Języki tworzono bez dostatecznej podstawy teoretycznej i często do użytku *ad hoc*.

Na początku 1963 roku występowało około 290 języków, z tego i

a) 74 zorientowanych maszynowo,

b) 56 przestarzałych (wycofanych),

c) 108 do obliczeń matematycznych,

d) 16 do zastosowań administracyjno-ekonomicznych,

e) 6 do zastosowań algebraiczne-ekonomicznych,

f) 7 do symulacji procesów technologicznych,

W owym czasie istniało 39 translatorów języka COBOL.

W 1966 roku naliczono już 1200 języków automatycznego programowania, nie licząc podzbiorów ALGOLu, COBOLu itp. Pojawił się również uniwersalny język programowania PL/I [1] .

[1] *W 1969 roku na publicznym forum – na Spring Joint Computer Conference w USA – pojawił się język BCPL (Basic Combined Programming Language) autorstwa Martina Richardsa z Cambridge. Język ten został zdefiniowany w 1966 roku i pierwszy kompilator opracował autor rok później. BCPL dał podstawy pod najważniejszy nurt programistyczny, jakim jest język C i jego pochodne. Cechy, które były preferowane do tego czasu – łatwość programowania i samodokumentowanie (np. w COBOLu) wynikajace z bliskości do naturalnego języka, ustąpiła miejsca precyzji myślenia i zwartości kodu. Język ten określany jest jako wieloparadygmatowy: strukturalny, imperatywny, proceduralny oraz posiadający tylko jeden typ danych (słowo – czyli stała liczba bitów). W efekcie doszło do uproszczenia pracy kompilatorów i przyśpieszenia przetwarzania. (Przyp.2010)*

<29>

IV. Klasyfikacja języków programowania

Do klasyfikowania języków programowania używać można wielu kryteriów. Niektóre z nich podano w niniejszym opracowaniu. I tak można te języki klasyfikować na języki:

1. wg techniki tłumaczenia
   a) interpretujące (interpretatory, interpretery, ang. *interpreters*),
   b) kompilujące (kompilatory, kompilery, ang. *compilers*),

2. wg poziomu programowania
   a) wewnętrzne (maszynowe),
   b) mnemoniczne operujące na adresach cyfrowych i stosujące symbole literowe do oznaczania kodu operacji,
   c) mnemoniczne pełne (symboliczne, ang. *assembly languages*) posiadające symbole literowe operacji i argumentów),
   d) symboliczne z makrorozkazami (o współczynnika tłumaczenia np. 4:1),
   e) wyższego rzędu, tj. nie zorientowane maszynowo.

3. wg funkcji
   a) do przetwarzania informacji,
   b) do obsługi przetwarzania (systemy operacyjne),
   c) do pisania translatorów,
   d) do opisywania języków (metajęzyki),
   e) do innych celów,

(Powyższe grupy klasyfikacyjne niekiedy zachodzą na siebie. W szczególności dotyczy to grup a, c, d).

4. wg pojęcia proceduralności – charakteryzujące się tym, że programista podaje kolejność procedur czy rozkazów

<30>

(do tego typu należą więc prawie wszystkie języki),

a) języki nieproceduralne tj. takie, w których programista podaje jedynie specyfikacje wejścia-wyjścia oraz wzory obliczeniowe, bez precyzowania sekwencji obliczeń (przykładem takiego języka może być REPORT PROGRAM GENERATOR).

Ze względu na ograniczoną objętość niniejszej pracy nie możemy omówić szczegółowo poszczególnych układów klasyfikacyjnych. Uwagę skoncentrujemy na językach do przetwarzania informacji.

ad 1. INTERPRETER charakteryzuje się tym, że natychmiast po dokonaniu tłumaczenia najmniejszej jednostki znaczeniowej programu (instrukcji) jest ona wykonywana, czyli otrzymujemy wynik. Jest to niewątpliwie zaleta z punktu widzenia użytkownika, wada zaś polega na tym, że trzeba dokonywać wielokrotnego tłumaczenia tych samych wyrażeń, np. przy obliczeniach cyklicznych.

KOMPILER tłumaczy najpierw cały program, a później dopiero po tzw. kompozycji (konsolidacji) program staje się gotowy do realizacji. Wprowadzenie zmian do programu pociąga za sobą konieczność rekompilacji.

Systemy interpretacyjne stosowane są z reguły w maszynach z małą pamięcią i dużą szybkością obliczeń. Istnieją języki łączące w sobie zarówno cechy interpretatora jaki kompilatora (np. QUIKTRAN). Większość języków pracuje w systemach kompilacji. Jako przykład specjalnego systemu interpretacyjnego wymienić można systemy operacyjne. Do systemu interpretacyjnego należą m.in. Języki MUMPS (*MGH Utility Multi-Programming System*) opracowany

<31>

na maszynę PDP-9 pod kątem wykorzystania wieloprogramowości oraz GOTRANy stanowiący wersję FORTRANu na IBM 1620. Język MUMPS wykorzystuje zaletę interpretatora polegającą na tym. że programy są krótkie, i w związku z tym nawet przy nie dużej maszynie można w pamięci przechowywać równocześnie kilka programów. Pozwala to na efektywne wykorzystanie właściwości podziału czasu (*time-sharing*) bez pośrednictwa pamięci dyskowej. Ponadto ułatwione jest uruchamianie programów, ponieważ istnieje możliwość bieżącej modyfikacji bez przebiegów rekompilacyjnych.

ad 2. Niewątpliwie podstawowym kryterium podziału języków jest dla programisty poziom programowania. Pisanie programów w języku wewnętrznym w maszynach III generacji (typu IBM seria 360, System 4, Seria 1900) nie jest (względnie prawie nie jest) stosowane. Wpływa na to kilka czynników (niezależnie od powyżej omówionych zalet języków wyższego rzędu):

a) przydział adresów rzeczywistych w warunkach pracy wieloprogramowej realizowany jest nie przez programistę, lecz automatycznie przez system operacyjny,

b) obsługa różnorodnych urządzeń wejścia-wyjścia wymaga zastosowania wielu procedur specjalnych i rozkazów dostosowanych do współpracy z systemem operacyjnym, zaprogramowania których w programach użytkowych ("zastosowaniowych") za pomocą cyfrowych (wewnętrznych)

instrukcji nastręczałoby poważne trudności,

c) ponieważ łatwo jest o pomyłkę adresów i kodów operacji wyrażanych cyfrowo, oraz trudno je wykryć kontrolą formalną, uruchomienie złożonych programów trwałoby bardzo

<32>

długo, co oczywiście byłoby kosztowne (koszt pracy maszyny i programisty),

d) z powodu długiego szkolenia i dużej pracochłonności programowania, byłoby trudno w krótkim czasie obciążyć maszynę w dostatecznym stopniu.

Stąd też zamieniono adresy i kody operacji symbolami łatwymi do zapamiętania (nazwami mnemonicznymi) oraz wprowadzono tzw. makrorozkazy zwalniające programistę od pisania typowych procedur.

Struktura instrukcji języka symbolicznego (poza makrorozkazami) zbliżona jest do struktury rozkazu języka wewnętrznego:

Etykieta (adres) instrukcji bloku	nazwa operacji	nazwy argumentów (operandów) lub adresy względne	nr indeksu (B-rejestru)

Jeśli chodzi o nazwy (symbole) operandów, to występuje ich tyle, ile jest adresów w rozkazie wewnętrznym. Programista posiada dostęp do poszczególnych rejestrów. Są to cechy szczególne, nie występujące w językach wyższego rzędu. Jedynie makrorozkazy traktować można jako swoisty rodzaj wstawek języka wyższego rzędu (jak wiadomo, języki wyższego rzędu budowane są z makrorozkazów).

Oto przykłady języków mnemonicznych

a) język mnemoniczny niepełny – City and Guilds Mnemonic Code: *ADD 302,5* oznacza – dodaj zawartość komórki o adresie (302 plus zawartość B rejestru Nr 5),

b) mnemoniczny pełny (plus makrorozkazy)

<33>

– SAS (System Adresów Symbolicznych dla ZAM-41):

SKO G8	skocz do procedury z etykietą *G8*
PISZ, CZYTAJ	makrorozkazy (występują one również w języku wyższego rzędu SAKO)

– Assembler IBM 360:

SAMPLE START X, 0	start programu od adresu wzgl. *0*

STRT BALR 9, 0	załad. rej. *B* zawartością adres. *0* [1] *Krzysztof Bytnerowicz zwrócił mi uwagę, że "BALR 9,0 nie ładuje rejestru 9 wartością 0 lecz adresem nastepnej instrukcji" (Przyp.2011)*
*USING *, 9*	zdefiniuj rejestr 9 jako bazowy

– USERCODE dla Systemu 4:

MP WORK (6) PRICE (2)	mnóż *WORK* przez *PRICE*, wynik jest przechowywany w *WORK*
ED DEST (11), VALUE	pole *DEST* jest redagowane w ten sposób, że nieznaczące zera są zamieniane przez znaki "spacje" lub "blank" względnie gwiazdkę
OPEN	nazwa zbioru, nazwa zbioru ..., nazwa zbioru (do 16) jest to przykład makrorozkazu otwarcia zbioru powodującego sprawdzenie metryki zbioru, itp. (lub zapisanie)
GET	makrorozkaz przywołania rekordu ze zbioru wejściowego

ad 3. Języki do przetwarzania informacji można podzielić na:

a) uniwersalne, np. PL/I,

b) specjalizowane (problemowe), np. COBOL, FORTRAN, ALGOL,

c) wąskospecjalizowane, np. języki do:

– symulacji,

– przetwarzania struktur listowych (napisów, symboli),

– redagowania wydawnictw graficznych i obsługi display'ów,

– sterowania obrabiarkami,

<34>

– tłumaczenia tablic decyzyjnych itp.

Jeśli chodzi o języki do przetwarzania struktur listowych, to istnieje ich na świecie kilkadziesiąt, przy czym do najpopularniejszych należą: LISP, COMIT, SNOBOL. W Polsce opracowano języki LOGOL i EOL. Niektóre właściwości tych języków mogą być z powodzeniem wykorzystane do translacji z języka na język.

Specjalną grupę tworzą języki do operowania na formułach FORMAC (umożliwiają rozwijanie wielomianów, ich dzielenie): ALPAK, FORMULA, ALGOL, SAINT.

Do popularnych kwestii należy porównywanie języków do obliczeń

numerycznych (a więc ALGOLu, FORTRANu) z językiem do przetwarzania danych ekonomiczno-administracyjnych (czyli z COBOLem), języki pierwszej grupy stosują wyłącznie zapis algebraiczny do oznaczania operacji arytmetycznych, podczas gdy COBOL oprócz tego zapisu (stosowanego po instrukcji *COMPUTE*) zapewnia również instrukcje słowne w rodzaju *DIVIDE*, *MULTIPLAY*, *ADD*, *SUBTRACT*. Składnia języków jest całkowicie odmienna. W COBOLu wprowadzono rozdzielanie opisu danych od procedur, stosując dwa odrębne rozdziały: DATA DIVISION i PROCEDURE DIVISION. Ponadto specjalny rozdział służy do opisu i rezerwacji urządzeń (ENVIRONMENT DIVISION). Najważniejsze znaczenie z punktu widzenia translatora posiada rozdział identyfikacji IDENTIFICATION DIVISION. Tego rodzaju podział funkcjonalny nie istnieje w językach do obliczeń numerycznych, nastawionych na konstruowanie procedur i podających jedynie podstawowe informacje na temat danych (deklaracje *INTEGER*, *REAL*, *FORMAT*, *DIMENSION*).

Opis danych stosowany w COBOLu pozwala na operowanie nazwami zbiorów (w instrukcjach OPEN, READ), rekordów, pól grupowych

<35>

i pól elementarnych, a więc wyraża złożoność "montażową" (strukturę) danych administracyjnych. ALGOL i FORTRAN operują nazwami jednostkowymi (niepodzielnymi), dając możność stosowania wielokrotności pól jednorodnych (indeksowanych). Tę samą właściwość posiada zresztą również COBOL.

Z punktu widzenia alokacji danych język COBOL zapewnia z reguły bardziej efektywny program (dzięki możliwościom "pakowania" danych). Natomiast generuje dłuższe programy dla takich samych zadań (np. w porównaniu z FORTRANem).

Teoretyczne wersje ALGOLu pomijają sprawę formalnego opisu przydziału urządzeń wejścia-wyjścia oraz tzw. edytowania (redagowania) danych wynikowych, pozostawiając to producentom.

Efektem tego jest zmniejszenie stopnia zamienności programów pisanych dla różnych maszyn.

Oto parę instrukcji wydawniczych firmy ICL dla Systemu 4—50:

WRITE / 30, FORMAT / '/ 'ND' / ' / , X/

gdzie: *30* jest nr urządzenia tj. drukarki wierszowej, *ND* oznacza wydruk 2 cyfr, *X* nazwa pola),

PAGE /30,1/ oznacza zmianę strony,

NEWLINE /30,1/ skok do nowego wiersza.

Są to instrukcje "firmowe" bez odpowiednika w innych wersjach. Ponadto ALGOL nie uwzględnia sprawy rozpoznania stałego i zmiennego przecinka, pozostawiając to maszynie. W niektórych

wersjach REAL oznacza liczbę zmiennoprzecinkową.

ALGOL dla Systemu 4-50 dysponuje bogatymi możliwościami wydawniczymi, porównywalnymi do podobnych cech COBOLu, mianowicie w deklaracji FORMAT można m.in. używać następujących

<36>

symboli: *D* – cyfra, *N* – drukowanie "najstarszej" cyfry jeśli jest znacząca (tj. nie równa zeru), *S* – spacja, • – kropka dziesiętna, *+ −* – znaki arytmetyczne.

Przykłady redagowania:

opis	pole przed zredagowaniem	pole po zredagowaniu
7S / NDDD	*12*	*……+…… 12*
3D4S.5D	*123.456789*	*123…….45678*

Również języki wzorowane na ALGOLu wprowadzają czasem własne środki opisu danych, jak np. ALGEM (*ALGOritmiczeski jazyk dla programmirowania Ekonomiczeskich i Matimaticzeskich zadacz*):
9 – cyfra w systemie dziesiętnym, *7* – cyfra w systemie ósemkowym, *1* – cyfra w systemie dwójkowym, *C* – dowolny znak, *A* – litera rosyjskiego alfabetu, *L* – litera łacińskiego alfabetu. • – kropka dziesiętna w postaci jawnej (dziurkowana), *Д* – kropka dziesiętna "założona", *П* – zamiana zer nieznaczących przez spację, ^ – znaki arytmetyczne.

<37>

Język ALGOL posiada dogodne właściwości operowania na zbiorach [1] jednorodnych danych, czyli masywach (*array*). Przynajmniej teoretycznie nie ma tutaj ograniczeń co do ilości (głębokości) indeksów i możliwy jest dynamiczny podział pamięci (granice zbioru mogą być określone wzorami arytmetycznymi).

[1] *W czasach kiedy pisałem książkę, "plik" nie był terminem powszechnie stosowanym. (Przyp.2010)*

Firmowe wersje ALGOLu wychodzą czasem dalej niż wersje formalne (publikowane jako REPORT ALGOL). Dotyczy to nie tylko wejścia i wyjścia, gdzie m.in. chodzi o powiązania z systemami operacyjnymi, lecz również o możliwość przywołania segmentów napisanych w języku symbolicznym (np. ALGOL 4-50 przywołuje segmenty napisane w USERCODE).

Język COBOL jest szczególnie przydatny w maszynach II generacji bez systemu operacyjnego, a to z tego względu, że operuje odpowiednim

aparatem formalnym do opisu konfiguracji maszyny niezbędnej do wykonania zadań, umożliwia automatyczne sprawdzanie (i zapisywanie) metryk oraz deklaruje przydział urządzeń we-wy dla poszczególnych zbiorów.

Języki do obliczeń numerycznych są prostsze, a więc i łatwiejsze do wyuczenia się. Opracowuje się do nich krótsze translatory i w związku z powyższym można ich używać nawet w maszynach z małą pamięcią operacyjną (8-16K). Wdrożenie pierwszych kompilatorów na małych maszynach napotykało na spore trudności. Wyrazem tego były przypadki, kiedy do skompilowania programów trzeba było ponad 40 przebiegów [A1].

Mimo sztywności programu (obowiązkowe sekcje, rozdziały) i tłumaczenia go techniką kompilacyjną, język COBOL jest niekiedy stosowany w systemach abonenckich.

<38>

Wg danych pewnej ankiety ([S4] 1969 r.), którą objęto 497 użytkowników w/w systemów, struktura używanych przez nich języków programowania była następująca:

FORTRAN	39.6%
BASIC	31%
COBOL	7,6%
PL/I	7,0%
ALGOL	2,8%

Jeśli chodzi o język BASIC (*Beginner's All-purpose Symbolic Instruction Code*), to opracowano go w Dartmouth College (w okresie 1964-1965) i wdrożono na maszynie GE 225. Pomyślany był, jako język do wstępnej nauki programowania przed przystąpieniem do ALGOLu lub FORTRANu. Posiada proste konstrukcje formalne, będąc w ten sposób bardzo łatwy do wyuczenia (w zasadzie programuje się już po pierwszym dniu nauki). Uruchamianie programów jest ułatwione. W dowolnej chwili można wprowadzać poprawki, podając jedynie nr sekwencji (zdania) i nowe jej brzmienie. Jako inne języki konwersacyjne wymienić można AMTRAN, JOSS, CAL, QUIKTRAN. Częściowo słusznym zarzutem (z dokumentacyjnego punktu widzenia można mu się przeciwstawić) w stosunku do COBOLu jest zbytnia obfitość części opisowych, jakiej musi używać programista. Szczególnie odnosi się to do opisu danych, wyrażania operacji arytmetycznych (jeśli brak, instrukcji *COMPUTE*). Wada ta daje się częściowo usunąć, jeśli stosowane są instrukcje COPY (do ściągania z biblioteki zarówno opisu danych jaki procedur, wspólnych dla kilku programów), oraz dozwolone jest użycie takich skrótów, jak *PIC* do oznaczenia *PICTURE*, *COMP* zamiast *COMPUTATIONAL* itp.

Znane są specjalne wersje COBOLu, używające jedynie minimalnych oznaczeń. Na przykład wersja COAX stosuje jednoliterowe

<39>

oznaczenia:

F' oznacza

FILLER (w *DATA DIVISION*) lub

FROM (w *PROCEDURE DIVISION*)

A' *ADD*

G' *GREATER* itp.

Inne usiłowania polegają na zastosowaniu specjalnych formularzy z gotowymi nadrukami (np. *RAPIDWRITE*).

Werbalność COBOLu związana jest niewątpliwie z celem, Jaki był stawiany przed tym językiem, kiedy chodziło o zbliżenie specyfiki programowania do specyfiki problemu (a więc do ludzi pracujących w określonym zawodzie).

Ponadto okazało się, że z pozoru proste tematy administracyjne są bardzo pracochłonne w programowaniu i należało stworzyć programiście takie narzędzie pracy, które pozwala szybko uruchamiać bardzo duże programy (to znaczy łatwo je analizować i szybko odnajdywać błędy) i będzie równocześnie stanowić dobrą dokumentację do kontynuacji pracy przez innych programistów.

COBOL bardziej uwzględnia wymogi nowoczesnej technologii przetwarzania, wykorzystując np. walor pracy równoczesnej (*overlapping*), dzięki możności podwójnego buforowania każdego wejścia i wyjścia oraz obowiązkowemu wprowadzenia odrębnych obszarów wejściowych i wyjściowych.

Pod względem czasu przetwarzania zadań angażujących duże zbiory i złożone rekordy język COBOL zdecydowania wyprzedza języki do obliczeń numerycznych (w tym również FORTRAN posiadający jakie takie możliwości opisu wprowadzanych danych – za pomocą deklaracji *FORMAT*).

<40>

Oto wyniki pewnych testów [J1] przeprowadzonych na maszynie IBM 7094 w połowie lat 60-tych:

	zadanie	COBOL	FORTRAN II

1.	Czytanie zbiorów i sprawdzanie jakości pól rekordów		
	a) wielkość bloku 10 rek.	2,3 min.	15 min.
	b) wielkość bloku 1 rek.	4,8	10,2
2.	Kopiowanie zbioru rozmiar bloku 10 rek.	2,8	13,8

Można powiedzieć, że wraz ze wzrostem pojemności pamięci i szybkości działania maszyn wzrasta efektywność kompilatorów COBOLu. Tam, gdzie tego nie udaje się osiągnąć w trakcie translacji, stosowane są specjalne optymalizatory, które przerabiają wersję otrzymaną w wyniku tłumaczenia. Przykładem takiego optymalizatora jest COBOL Optimiser opracowany w Capex Corp. of Phoenix Arizona [X2].

Nie można mówić o efektywności kompilatora w sensie ogólnym. Uważa się, że dobre kompilatory COBOLu opracowuje firma Honeywell, o czym świadczy fakt, że udział wdrożonych kompilatorów tej firmy jest znacznie wyższy od udziału w sprzedaży maszyn.

Wydaje się, że na temat przydatności COBOLu przede wszystkim powinni wypowiadać się użytkownicy. Otóż wg ankiety [X3] przeprowadzonej w USA wśród 724 użytkowników COBOLu, 560 z nich

<41>

(ok. 80%) potwierdziło pełną przydatność tego języka, przy czym 65% z tej grupy oświadczyło, że zakres stosowania tego języka stale się u nich zwiększa.

Prowadząca ankietowanie Auerbach Corp. podała wyniki mówiące, że użycie COBOLu jest dwukrotnie wyższe w takich zastosowaniach, jak ewidencja zapasów, rozliczenia zakupów, sprzedaży itp. zaś język symboliczny dorównywał COBOLowi jedynie w przypadkach przetwarzania wyrywkowego (*random access job*).

Największymi zwolennikami COBOLu były najczęściej ośrodki małe, zatrudniające do kilkunastu programistów [1] .

Reasumując zalety COBOLu stwierdzić można, że jest to język stosunkowo łatwy do wyuczenia i stosowania (w porównaniu z językami symbolicznymi) oraz przedstawiający duże walory dokumentacyjne. Dzięki zastosowaniu zwrotów *PERFORM DEPENDING ON*, itp. upraszcza się organizacja programu, zaś w czasie testowania pomocne mogą być procedury *TRACE*, *EXHIBIT* itp.

Wadami COBOLu, w porównaniu z językami symbolicznymi, są.

a) większa, np. o 30%, zajętość pamięci operacyjnej,

dłuższy, np. o 10%, czas przetwarzania [T2],

zaś zaletami: kilkakrotne zmniejszenie pracochłonności programowania i czasu uruchamiania programów.

Skoro mówimy o językach do przetwarzania danych, nie możemy pominąć arcyciekawej próby (1959, USA) stworzenia "algebry informacji" w postaci języka LSG (*Language Structure Group*).

[1] *Na poparcie danych literaturowych mogę od siebie dodać, że w czasie mego 7-miesięcznego pobytu (1970-71) w Wielkiej Brytanii spotkałem ośrodki obliczeniowe, w których posługiwano się prawie wyłącznie językiem COBOL.*

<42>

Znana są próby zastosowania tego języka do obliczania list płac.

Analiza porównawcza języków programowania jest utrudniona, ponieważ stale ulegają one zmianom, zarówno w wersjach formalnych, jaki firmowych. Obserwuje się m. in. wzajemne zapożyczenia, istnieją grupy języków wzorowane na FORTRANie, ALGOLu i COBOLu, względnie czerpiące ze wszystkich tych języków na raz. Przykładem wpływu FORTRANu na ALGOL może być deklaracja *FORMAT* używana w niektórych wersjach ALGOLu po instrukcji *WRITE* (to też niestandardowa instrukcja) oraz instrukcja *READ* z numerem urządzenia jako parametrem.

Niektóre języki (czy też raczej pakiety programowe) posługują się wprost aparatem formalnym innego języka, jak to jest w przypadku BASEBALL (system pytanie-odpowiedź), używającego zwrotów języka IPL-V do przetwarzania struktur listowych.

Wyrazem pewnego podsumowania dorobku programowania wydaje się być uniwersalny język PL/I. Stanowi on zjawisko ze wszech miar godne uwagi. Nie stał się jeszcze jednak językiem powszechnego stosowania (w 1968 roku tylko 1% komputerów miało opracowany translator tego języka – wg [R3]). W celu zbadania efektywności tego języka w zastosowaniach ekonomiczno-administracyjnych przeprowadzono testy porównawcze z COBOLem na przykładzie konkretnych programów m.in. dotyczących list płac [R3]. Porównanie to nie podważyło pozycji COBOLu.

Oto wyniki:

a) uruchamianie programów PL/I trwało znacznie (dwukrotnie) dłużej, mimo krótszych programów zewnętrznych,

<43>

b) czas pisania programów PL/I był tylko nieznacznie krótszy,

c) czas przetwarzania był w przypadku COBOLu znacznie krótszy (dla dwóch programów wynosił 2/3 i 1/3 czasu PL/I),

d) uczenie się języka PL/I jest znacznie trudniejsze, natomiast później użycie aparatu formalnego wydaje się być łatwiejsze, jest to więc niejako język przeznaczony dla zawodowych programistów, specjalizujących się w programowaniu PL/I (wtedy rekompensują się nakłady na szkolenie),

e) PL/I zapewnia lepsze możliwości kontroli logicznej i operowania na danych (włącznie z manipulacją na bitach, nie zawsze stosowaną w COBOLu). Natomiast COBOL posiada lepsze procedury czytania i pisania (włącznie z takim ułatwieniem wydawniczym, jakim jest REPORT-WRITER),

f) łatwiejsza jest korekta czy też modyfikacja programów PL/I.

Tyle mówią praktyczne konfrontacje. Natomiast pod względem poziomu programowania, język PL/I przewyższa pozostałe języki. Jest próbą opracowania języka adekwatnego do hardware'u i systemu operacyjnego maszyn III generacji.

Język PL/I opracowano, starając się uwzględnić osiągnięcia istniejących języków, a w szczególności FORTRANu, ALGOLu i COBOLu.

Notacja języka PL/I jest stosunkowo zwięzła i w tym względzie PL/I opiera się raczej na tradycjach FORTRANu i ALGOLu, niż COBOLu. Język ten uważany jest za szczytowe osiągnięcie języków proceduralnych. Ponieważ ma być samowystarczalny, nie przewiduje on wstawek z innych języków (przynajmniej nie przewidywały tego pierwsze wersje).

<44>

Specyficzną cechą PL/I jest powiązanie z systemem operacyjnym i hardware'm. Wyraża się to przede wszystkim w organizowaniu pracy równoległej, zarówno w przetwarzaniu wielomaszynowym jak i w maszynie wieloprogramowej. Przykładem tego rodzaju pracy jest tzw. wieloproceduralność (lub podprogramowanie), polegająca na tym, że równolegle wykonuje się kilka procedur tego samego programu, przy czym zorganizowane może to być asynchronicznie lub w systemie przerwań.

Asynchroniczność osiągana jest m.in. poprzez stosowanie takich

instrukcji, jak wstrzymanie procedury *WAIT*, przerwanie procedury na określoną ilość jednostek czasu *DELAY*, badanie ukończenia *COMPLETE* itp.

Z punktu widzenia użytkownika specjalne znaczenie posiadają dwie cechy PL/I: modułowość i zasada domyślności. Modułowość sprawia, że nie trzeba znać języka w całości, lecz stosuje się jego podzbiory stosownie do poziomu programisty i złożoności zastosowania.

Dzięki zasadzie domyślności, w przypadku wykrycia braku deklaracji translator nie sygnalizuje błędu, lecz dobiera deklarację najbardziej prawdopodobną w danym przypadku. Na przykład, jeśli przy zmiennej *EXTERNAL* nie ma deklaracji sposobu przydziału pamięci, translator przyjmuje *STATIC*.

Bardzo zwięzły w porównaniu z COBOLem jest sposób opisu struktury danych (połączony od razu z indeksowaniem):

DECLARE 1 A(2), 2B, 2C oznacza:		
poziom 1	*A(1)*	1-sze wystąpienie *A*
	2B	
	2C	
1	*A(2)*	2-gie wystąpienie *A*
	2B	
	2C	

<45>

W PL/I rozróżnia się dwa sposoby przesyłania danych:

a) zorientowany nt. tzw. strumień,

b) zorientowany na rekord (zapis, dokument).

Do wejścia-wyjścia strumienia używa się instrukcji *GET* i *PUT*, zaś w drugim przypadku *READ* i *WRITE*. Przy strumieniowym przesyłaniu dane są rozpatrywane jako ciągły strumień elementów w postaci znakowej. W rekordowym przesyłaniu jednostką przesyłową jest rekord, przy czym jego elementy mogą być zakodowane w różnych formach.

Głównymi pojęciami opisu danych są maszyny i struktury. Przyrównując je do tradycyjnych pojęć, struktura odpowiada rekordowi, zbiorowi (jako złożona struktura), zaś masyw – tabeli jednorodnych danych, np. macierzy.

Sposób kompilacji programów PL/I różni się znacznie od dotychczasowych rozwiązań. Mianowicie obejmuje również elementy

przetwarzania (np. obliczanie wspólnych lub jednorazowych wyrażeń) oraz umożliwia programiście włączenie się czynne do procesu kompilacji.

Języki wąskospecjalizowane

Języki te służą do programowania takich specjalnych zastosowań, jak:

a) sterowanie obrabiarkami (machine tool control) np. APT,

b) konstrukcje logiczne np. LOTIS, APL,

c) budownictwo cywilne np. COGO, STRESS,

d) pisanie translatorów np. CLIP, COGENT,

e) symulacja cyfrowa np. GPSS, SIMSCRIPT, SOL,

f) systemy pytanie-odpowiedź np. COLINGO, BASEBALL, QUERY, ADAM, DEACON,

g) wyjścia graficzne i ekranowe np. GRAF, PENCIL, DOCUS,

h) tłumaczenie tablic decyzyjnych.

Przeznaczeniem języków wąskospecjalizowanych jest obsługa wybranego rodzaju zastosowania, w którym są łatwiejsze w użyciu i efektywniejsze od innych. Wynika to z wyposażenia tych języków w specjalistyczne instrukcje, jak np.

a) w APT : *SPINDL/OFF* (*turnoff spindle* – obtoczyć wał)

TL RGT, GO PGT/BASE (*with the tool on the right go right along the line BASE*)

b) COLINGO: *UPDATE, CHANGE, EXECUTE, ALLOCATE, ANALYZE*.

Skoro wspomnieliśmy o języku APT, warto dodać, że opracowany został w MIT (*Massachusetts Institute of Technology*) w latach 1952 – 1956. Ulepszone wersje to: 1958, APT II, 1961 APT III.

Jeśli chodzi o język do symulacji, to jednym z pierwszych języków tego

typu był język DYNAMO opracowany w MIT w 1959 roku dla IBM 704. Najbardziej znanymi językami w zakresie symulacji są GPSS, SIMSCRIPT i SOL.

Opis GPSS (*General Purpose SImulation System*) po raz pierwszy opublikowano w 1961 roku. SIMSCRIPT (*Simulation Oriented Language*) został opracowany w RAND CORP. w 1960 roku. SOL (*Simulation Oriented Language*) stanowi próbę połączenia dwóch różnych systemów językowych, a mianowicie systemu "blokowego" (GPSS) i "zdaniowego" (SIMSCRIPT). Języki do pisania translatorów można podzielić na dwie grupy:

<47>

1. oparte o zasady zwykłych języków programowania, np. CLIP (*Compiler Language for Information Processing*) oparty został na ALGOLu 58, lecz posiada istotne uzupełnienia w zakresie manipulacji na danych (np. podanie rozmiaru nieindeksowanych i danych) oraz instrukcje wejścia-wyjścia,

2. do tzw. tablicowego tłumaczenia składni (*syntax directed table-driven compilation*), np. COGENT (*Compiler and GENeralized Translator*) opracowany dla CDC 3600.

W celu pokazania specyfiki języków do pisania translatorów przytoczymy przykład instrukcji w języku TMG:

INTEGER... ZERO-MARKS DIGIT INSTALL co oznacza: pomiń nieznaczące zera, zaznacz początek łańcucha i znajdź co najmniej jedną cyfrę, a następnie wyspacjuj wszystkie następne cyfry oraz umieść "symbol" do tabeli znaków i drzewa (grafu) wyjścia.

Generator programów

Przez generację rozumiemy automatyczne tworzenie typowych (powtarzalnych) sekwencji operacji w oparciu o "skrótowe" dane (rodzaj sekwencji, parametry) dostarczone przez programistę.

Ze względu na radykalne zmniejszanie pracochłonności programowania, zastosowanie generatorów wydaje się mieć duże perspektywy. Wyróżnić można trzy podstawowe poziomy generacji:

a) makrogenerator,

b) generator programu standardowego,

c) generator-język programowania,

d) makrogenerator

Generuje rozkazy elementarne na podstawie makrorozkazu i deklaracji (opisu danych itp.) umieszczonych w programie zewnętrznym [1] .

[1] *Inną drogą realizacji makrorozkazu z pominięciem generacji jest przywołanie gotowego podprogramu.*

<48>

Makrogeneratory są szczególnie przydatne do wyrażania operacji o wielu możliwych kombinacjach (np. typów operacji dodawania, zależnych od charakteru danych, żądanych zaokrągleń, kontroli formatu wyniku itp.). Kody tych operacji są każdorazowo generowana.

b) Generator programu "standardowego"

[2] *Pierwsza na pomysł generowania programu przez inny program wpadła Francis Holberton w 1949 roku i wykorzystała to potem do geneerowania programów sortowania i łączenia. [Przyp. 2011]*

Służy do generowania takich programów, jak np. SORT, REPORT, WRITKK, które w klasycznej postaci posiadają sztywną formę programów standardowych. Zaletą metody jest wyeliminowanie ograniczeń programu standardowego i wyższa efektywność. W przypadku generacji SORT, generator na podstawie parametrów i zasadniczego szkieletu działania programu generuje program sortowania dopasowany do konkretnych wymogów (rozmiarowi typów rekordów, rodzaju i ilości kluczy sortowania itp.) oraz dostępnej konfiguracji. Jeden z pierwszych generatorów SORT został napisany przez [wyżej wspomnianą] F.Holberton jeszcze w pierwszej połowie lat 50-tych [S1].

REPORT-WRITER służy do wybierania danych ze zbiorów oraz organizacji Ich wydruku. Istnieją również generatory do aktualizacji zbiorów, generowania kompilatorów (tzw. *compiler-compiler*, *compiler generator*) itp.

c) generator – język programowania

Jako przykład tego rodzaju (nieproceduralność !) języka programowania wymienić można RPG-REPORT PROGRAM GENERATOR, opracowany m.in. dla maszyn IBM 1401, seria 360, UNIVAC.

<49>

RPG umożliwia proste zaprogramowanie nawet złożonego problemu z dziedziny przetwarzania danych administracyjno-ekonomicznych (obejmującego szereg operacji obliczeniowych, wydruki wg różnych

stopni kontroli itp.).

Programista posługuje się standardowymi arkuszami programowania o nadrukach odrębnych dla:

1. opisu zbiorów (rodzaje),

2. opisu struktury rekordów wejściowych,

3. opisu danych wyjściowych na drukarkę wierszową (pola, tytuły, sumy wg stopni kontroli, znaki wydawnicze),

4. opisu obliczeń (rodzaje operacji, sposób kontroli obliczeń).

Zwolennicy RPG twierdzą [R6], że 80-90% programów przetwarzania danych może być z powodzeniem napisane w tym języku. Optymiści [X4] oczekują, że w najbliższej przyszłości 75% wszystkich programów przetwarzania danych zostanie napisanych w RPG, podczas gdy tylko 15% przypadnie na COBOL a 10% na BAL i FORTRAN.

Systemy operacyjne (CONTROL PROGRAM, CONTROL LANGUAGE, CONTROL SOFTWARE, OPERATING SYSTEM)

Opracowanie systemów operacyjnych stanowi problem równie ważny jak rozwój "użytkowych" języków programowania. Chodzi przede wszystkim o zmniejszenie przestojów maszyny (poprzez zredukowanie do minimum czynności operatora, automatyczne przełączanie pracy maszyny z zadania na zadanie, harmonogramowanie pracy urządzeń) oraz stworzenie języka komunikacji operatora z maszyną. Wymogi systemów operacyjnych coraz bardziej rzutują na konstrukcję języków programowania.

<50>

System operacyjny jest niezbędny w maszynach wieloprogramowych, a znaczenie jego szczególnie wzrasta w wielomaszynowym przetwarzaniu. Dzięki niemu uzyskać można tzw. POLIMORFICZNOŚĆ, polegającą na tym, że zestaw komputerowy zmienia swą "postać" stosownie do problemów, poprzez zmianę połączeń pomiędzy elementami składowymi konfiguracji oraz wprowadzanie zmian strukturalnych (zmiana długości słowa, akceptowanie innej listy rozkazów itp).

Pierwsze publikacje na temat systemów operacyjnych pojawiły się w 1953 roku i dotyczyły systemów SOS (*Share Operating System*) i FMS (*Fortran Monitor System*) opracowanych dla maszyn IBM 709 i 704 [T2].

Firma Honeywell pierwszy swój system operacyjny opracowała w 1957 roku dla maszyny D-1000, zaś w 1960 roku stworzyła pakiet EXECUTIVE do kierowania pracą wieloprogramową (do 7 programów [X5]).

Proste systemy operacyjne (zwane w IBM dyrygentami, w odróżnieniu od szerszego pojęcia systemu operacyjnego), składające się z kilkuset rozkazów, służyły głównie do wzywania standardowych podprogramów umieszczonych na taśmie bibliotecznej. Przykładem takiego dyrygenta jest MONITOR-70 dla maszyn IBM 7070/7074 [F1].

Nieco większe systemy operacyjne sterowały również pracą urządzeń wejścia-wyjścia oraz realizowały kontakt operatora z maszyną (i odwrotnie). Systemy operacyjne maszyn III generacji kierują pracą wieloprogramową, przydzielają urządzenia wejścia-wyjścia, dokonują przydziału obszarów pamięci operacyjnej itp.

Przez pojęcie systemu operacyjnego firma IBM i firma Honeywell rozumieją cały kompleks oprogramowania sterującego. Przybliżeniem do tej koncepcji był system MONITOR MACDONALD dla maszyn IBM

<51>

709/7090, obejmujący swym zasięgiem kierowanie tłumaczeniami. Jedną z pierwszych wersji systemu operacyjnego IBM był SO dla IBM 1410/7010, zaś jej pełnym wcieleniem – oprogramowanie serii 360.

Analiza porównawcza systemów operacyjnych jest trudna, ponieważ każda wersja operuje innym aparatem pojęciowym, zaś pod te same pojęcia podkłada inną treść funkcjonalną. Ogólnie rzecz biorąc, system operacyjny jest to zbiór "prawideł" (programów), czyniący komputer zdolnym do pracy.

System operacyjny serii 360 składa się z programów sterowania (*Control Programms*) i programów przetwarzania. Głównym jego celem jest zwiększenie wydajności komputera. Np. *Management Task* grupuje zadania w takiej kolejności, która zapewni lepsze wykorzystanie urządzeń. Poniżej podamy zwięzłą charakterystykę systemów operacyjnych serii 360 i maszyn firmy Honeywell. Następnie spróbujemy dokonać analizy porównawczej następujących systemów: IBM, ICL seria 1900, ICL system 4, NCR Century, w celu zilustrowania istniejących rozbieżności.

<52>

Tabela 1. **System operacyjny IBM 360**

PROGRAMY STEROWANIA	PROGRAMY PRZETWARZANIA
Kierowanie przepływem danych [1]	Translatory: - PL/I - FORTRAN - ALGOL - COBOL - język symboliczny - RPG - telekomunikacja - generatory programów - generator systemu - translator (*debugger*) testu - translator tekstu
(*Data Management*)	
Kierowanie przetwarzaniem zestawu zadań (*Job Management*)	
Kierowanie przetwarzaniem zadań	Programy usługowe: (*service programms*) - program łączącо-redagujący (*linkage editor*) - *sort merge* - programy standardowe (*utilities*)
	Programy problemowe użytkowników

[1] *Termin data management bywa tłumaczony jako: "operowanie danymi", "sterowanie danymi" lub "sterowanie ruchem danych" (Przyp.2010).*

<53>

Tabela 2. **System operacyjny Honeywell OS/200<**/p>

I. SUPERVISOR

1. MONITOR STAŁY (*Resident Monitor*)
- wprowadzanie programów
- kierowanie pracą wieloprogramową jedno- lub dwustrumieniową
- kierowanie wejściem-wyjściem: alokacja kanałów przesyłowych i urządzeń
- kierowanie konwersją danych: karty -- taśma magnet., taśma magnet. -- drukarka, taśma magnet. -- dziurkarka taśmy
- kierowanie komunikacją

2. MONITOR PRZEJŚCIOWY (*Transitional Monitor*)
- wczytywanie kart sterujących następnego programu i przekazanie parametrów do monitora stałego
- inicjowanie komunikacji
- wykonywanie dumpingu (PAO, dysku, taśma magnet.)

3. KIEROWANIE ZBIORAMI BAZY DANYCH
(Data Base File Management)
składnik opcjonalny

II. KOMPONENTY ZALEŻNE (*Dependent Components*)

1. PODSYSTEM KIEROWANIA DANYMI (dla zbiorów w pamięci dyskowej)
- kierowanie dostępem, założenie i aktualizacja rozmieszczenia zbiorów, zamiana ścieżek, załadowanie zbiorów, listowanie rozmieszczenia (map)
- podprogramy wejścia-wyjścia: pakiety logiczny i fizyczny we-wy, itp.

2. PODSYSTEM JĘZYKÓW I PROGRAMÓW
- translatory, archiwowanie programów źródłowych i wewnętrznych

3. PROGRAMY STANDARDOWE

<54>

Tabela 3. **Zestawienie porównawcze systemów operacyjnych**

IBM BOS, DOS, TOS	ICL seria 1900	ICL system 4	NCR Century
1. IPL (*Initiate Program Loader*) program inicjujący wprowadzanie 2. SUPERVISOR - obsługa przerwań - sterowanie wejściem i wyjściem - sygnalizowanie błędów - koordynacja przejścia od *Ask Control* do *Job Control* 3. JOB CONTROL	1. EXECUTIVE - kontakt z operatorem - ładowanie programów - przydzielanie urządzeń - organizacja pracy wieloprogramowej 2.a. AUTOMATYCZNY OPERATOR 2.b. GEORGE (*General Organisational Environment*) stanowi rozwinięcie funkcji EXECUTIVE GEORGE IV: - stronicowanie - harmonogra- mowanie prac - rejestracja wykorzystania jedn. centr. i urządzeń we-wy przez poszczególne programy GEORGE III: zawiera m.in. MOP (*Multiple On-line Programming Module*) umożliwiający pracę w wielodostępie dla 60 abonentów	1. EXECUTIVE a) Job Control - przydział pamięci - przydział urządzeń - wprowadzanie programu (*Job Input*, *Job Scheduler*, *Allocator*, *Deallocator*) b) Supervisor - kontakt z operatorem - system przerwań - kronika błędów - nadzór nad pracą wieloprogram. (S05J – 6 strumieni, 14 programów) 2. PROGRAM TRIALS SYSTEM 3. UTILITIES (program standard.)	1. MONITOR - wprowadzanie programu - kierowanie kolejnością pracy 2. INPUT-OUTPUT EXECUTIVE - przydział pamięci - przydział urządzeń wejścia-wyjścia - kontrola błędów 3. KIEROWANIE PRZEPŁYWAMI DANYCH (*Data Traffic Control*) - harmonogra- mowanie - system przerwań

Jak wynika z powyższego, rozbieżności dotyczą szczególnie takich pojęć, jak SUPERVISOR i EXECUTIVE, które w każdym przypadku znaczą właściwie co innego przy zachowaniu pewnych wspólnych funkcji.

<55>

Ogólnie rzecz biorąc system operacyjny spełnia następujące funkcje:

– wprowadzenie programów i kontrola ich realizacji,

– interpretacja rozkazów operatora i ich wykonanie,

– raportowanie odchyleń operatorowi (np. nadmiaru, przekroczenia limitu pamięci itp),

– interpretacja kart sterowania (CONTROL CARDS, zwane też *Steering Cards*),

– harmonogramowanie przebiegów,

– kontrola pracy urządzeń peryferyjnych,

– przydział urządzeń peryferyjnych do programów i zwalnianie tych urządzeń,

– zatrzymanie programu, o ile dalsza jego realizacja nie może być kontynuowana oraz powtórne uruchomienie po usunięciu przyczyny zatrzymania,

– kierowanie pracą wieloprogramową.

Złożoność systemu operacyjnego SO stale wzrasta np. SO dla maszyny IBM 360-91 zawiera ponad półtora miliona instrukcji. GEORGE III wraz z EXECUTIVE zajmuje około 12 tysięcy słów pamięci operacyjnej.

System operacyjny MASTER (CDC 3300) stosowany jest w maszynach wyposażonych w co najmniej 64K PAO. Dzięki temu systemowi, uzyskano tak sprawną pracę wieloprogramową, że przepustowość maszyny zwiększyła się o 40% [X6].

System operacyjny SIPROS (*SImultaneous PRocessing Operating System*) dla CDC 6000 stanowi przykład kierowania bardzo złożoną konfiguracją 11 jednostek przetwarzania, w której występuje równoczesność. wykonywania programów i instrukcji, dostępu do poszczególnych modułów pamięci operacyjnej oraz wykonywana jest

<56>

tzw. rekonfiguracja (dodanie nowego składnika, usunięcie uszkodzonego).

V. Geneza języków powszechnego zastosowania

Pisząc o językach powszechnego zastosowania mamy na myśli nie ich uniwersalność, lecz liczbę wdrożeń translatorów i uruchomionych programów.

Na pewno za takie języki można uważać ALGOL, FORTRAN i COBOL, zaś język PL/I (jako uniwersalny i nowoczesny) wydaje się mieć szanse zostania takim językiem. Dużą przeszkodą w osiągnięciu powszechności języków programowania były zaściankowe interesy firm. Po pierwszym okresie przykrych doświadczeń specjaliści dochodzą do wniosku, że znaczenie języków programowania wykracza daleko poza rolę narzędzia walki konkurencyjnej i kwestia jakości

(oraz powszechności) języków nie jest sprawą wewnętrzną jednej firmy. W ten sposób doszło do powstania języków ALGOL i COBOL, jako produktu współpracy przedstawicieli firm i naukowców.

1. **Geneza ALGOLu**

ALGOL (ALGOrythmie Language) został opracowany przez przedstawicieli Anglii, Danii, Francji, Holandii, NRF, Stanów Zjednoczonych i Szwajcarii. Wersję języka przygotowano na szeregu spotkań odbytych na terenie Europy i Stanów Zjednoczonych.

Największe zasługi wydają się mieć dwie organizacje:

<57>

GAMM (Geselschaft für Angewandte Mathematik und Mechanik) oraz ACM (Association for Computing Machinery). GAMM już w 1955 roku powołał zespół do prac nad sprawami translacji wzorów matematycznych. W 1957 roku zawarte zostało porozumienie pomiędzy ACM i GAMM w sprawie opracowania wspólnego języka.

Grupa Europejska składała się z przedstawicieli organizacji GAMM, Association Francaise de Calcul, British Computer Society i Nederlands Rekenmachine Genootschap. W skład grupy Amerykańskiej wchodziły organizacje: ACM, USE, SHARE i DUO.

W roku 1958 nastąpiło w Zurichu spotkanie obu tych grup. Na łamach *Communications of the ACM* postanowiono drukować rezultaty prac. W 1958 roku opublikowano pierwszą wersję ALGOLu, zwanego wówczas IAL (*International Algebraic Language*).

Wersję ALGOL 60 poprzedziło spotkanie w 1959 roku w Paryżu, na którym John Backus zaprezentował formalną metodę definiowania składni, nazwaną później BNF (*Backus Normal Form* lub *Backus Naur Form*) oraz przedstawił użycie tej metody do definiowania ALGOLu. Na następnym spotkaniu w Paryżu w tym samym roku Naur zreferował wersję ALGOL 60.

W 1962 roku ulepszono język poprzez usunięcie błędów i nie jasności tworząc Revised Report on the Algorythmic Language ALGOL 60. Wersja ta wraz z dodatkowymi specyfikacjami wejścia-wyjścia została zatwierdzona jako norma ISO.

Na wiosnę 1968 roku opublikowano opis ALGOLu 68 pod redakcją van Wijngaardena. Algol 68 zdefiniowany został przez Grupę Roboczą 2.1. IFIP. Wersja ta zawiera szereg nowych możliwości: przetwarzanie równoczesne (*parallel processing*), segmentowa kompilacja, możliwość deklarowania nowych struktur danych,

<58>

operowania na liczbach podwójnej i potrójnej precyzji, tablice o zmiennych granicach, priorytety, format danych itp.

Zasługi ALGOLu dla rozwoju teorii i praktyki programowania są znaczne. Przede wszystkim po raz pierwszy zaproponowany został język stanowiący formalną metodę definiowania składni, będący równocześnie językiem publikacji. ALGOL umożliwia proste i zwięzłe zaprogramowanie procedur obliczeniowych, "nieograniczone" indeksowanie elementów tablic, procedury rekursywne itp. Dzięki tym właściwościom ALGOL stał się podstawą konstrukcji szeregu języków problemowych;

Na bazie ALGOLu 58 opracowano języki: MAD, NELIAC, JOVIAL. Ten ostatni przerodził się w język o wielu dodatkowych możliwościach. Ponadto w oparciu o ALGOL powstały tak różnorodne języki problemowe, jak:

a) SIMULA (język do sterowania),

b) DIAMAG (*time-sharing system*),

c) LISP 2 (obliczenia algebraiczne, analiza lingwistyczna),

d) GPL (*General Purpose Language*),

e) AED (obliczenia konstrukcyjne),

f) SFD-ALGOL (*System Function Description* – ALGOL).

Niektóre wersje ALGOLu noszą inne nazwy np. SMALGOL (Small ALGOL), BAL GOL (Burroughs ALGOL).

Na szczególne podkreślenie zasługuje JOVIAL (*Jules' Own Version of the International Algebraic Language*), posiadający właściwość tzw. COMPOOL (COMmunication POOL) – pracy równoległej (jedyny język z tą właściwością do momentu pojawienia się PL/I) oraz możliwość użycia do pisania własnego kompilatora.

<59>

2. **Geneza FORTRANu**

FORTRAN (*FORmula TRANslator*) należy do najstarszych języków programowania. Dzięki kolejnym udoskonaleniom (FORTRAN II,

FORTRAN IV) ten firmowy język IBMu utrzymuje się do dzisiejszego dnia, jako jeden z najpopularniejszych języków programowania zarówno w USA Jaki w Europie. Język ten okazał pewien wpływ na pierwsze wersje ALGOLu (jeden z twórców ALGOLu J. Backus uczestniczył w opracowywaniu FORTRANu), zaś z kolei ALGOL oddziaływał na FORTRAN IV.

Pierwszy dokument na temat FORTRANu znany był już w roku 1954, zaś podręcznik powstał w okresie 1956-1957. W 1958 roku opracowano FORTRAN II, który różnił się od pierwszej wersji m.in. dodaniem END i możliwością przywoływania podprogramów (poprzez *CALL*, *SUBROUTINE*). Obie wersje były mocno ukierunkowane na sprzęt (*WRITE*, *TAPE*, *READ DRUM*).

W 1962 roku powstaje FORTRAN IV dla maszyny IBM 7030 (STRETCH), w której to wersji można używać deklaracji *LOGICAL*, *DOUBLE-PRECISION*, *REAL*, *INTEGER*, logicznej instrukcji *IF* (obok arytmetycznej) itp. oraz usunięte zostały wyrażenia związane z przełącznikami (kluczami) i słowa *TAPE*, *DRUM* z instrukcji *READ* i *WRITE*.

FORTRAN był jednym z pierwszych języków objętych standaryzacją przez ASA (*American Standards Association*, później USASI – *United States Standards Institute*, obecnie ANSI – *American National Standards Institute*).

W latach 1962-1964 opracowano w IBM język FORMAC (*FORmula MAnipulation Compiler*), jako uzupełnienie FORTRANu IV w zakresie

<60>

manipulacji algebraicznych. Później okazało się, że FORMAC z powodzeniem może współpracować z językiem PL/I.

Rozważając wpływ FORTRANu na inne języki należy dodać, że już pierwsze wersje tego języka (FORTRAN I, FORTRAN II) Stanowiły podstawę opracowania szeregu fortranopodobnych kompilatorów: FORTRANSIT (IBM 650), GOTRAN (1620), Honeywell Algebraic Compiler (H-800), ALTAC (Philco 2000), itp.

Do języków specjalnych wzorowanych na FORTRANie zalicza się m.in. QUIKTRAN (*on-line time sharing language*), GRAF (*GRaphic Addition to FORTRAN*), DSL/90 itp.

3. **Geneza COBOLu**

Opracowywanie COBOLu (*COmmon Business Oriented Language*) rozpoczęto później niż FORTRANu i ALGOLu.

Mimo dużej złożoności języka pierwszą wersję opracowano stosunkowo szybko, bo w przeciągu jednego roku. Głównymi inicjatorem COBOLu była Marynarka Wojenna USA, będąca jednym z największych

użytkowników maszyn matematycznych. Pierwsze spotkanie "zainteresowanych" odbyło się 8 marca 1959 roku w Uniwersytecie Pensylwania, zaś 28-29 maja zorganizowano naradę (40 osób) w Departamencie Obrony (DOD) USA. Uczestnikami narady byli przedstawiciele szeregu firm produkujących komputery m.in. IBM, Burroughs, Honeywell, Univac, RCA) oraz przedstawiciele organizatora i instytucji rządowych.

Problem polegał na tym, że instytucje rządowe (a w szczególności United States Navy) posiadały maszyny różnych producentów i istniały w związku z tym kłopoty z wymianą doświadczeń (kadr i oprogramowania) pomiędzy poszczególnymi, ośrodkami.

<61>

Zaproponowano więc opracowanie języka wspólnego dla różnych typów maszyn, przystosowanego do programowania problemów administracyjnych.

Na wspomnianej naradzie powołano komitety robocze CODASYLu (*Committee On DAta SYstems Languages*) a mianowicie: *Short*, *Intermediate* i *Long Range Committee*.

Prace nad COBOLem rozpoczęto od studiowania takich języków, jak FLOW-MATIC, AIMACO, COMTRAN (*COmercial TRANslator*). W późniejszych pracach uwzględniono niektóre cechy języka FACT. Już we wrześniu otrzymano pierwszą wersję języka zaś w styczniu 1960 roku została ona zatwierdzona przez Komitet Wykonawczy CODASYLu i w kwietniu opublikowana.

Trzeba podkreślić, że tak szybkie rezultaty osiągnięto przechodząc od dyskusji w szerokim gronie do intensywnej pracy małej liczby specjalistów (w końcowych pracach brało udział jedynie 6 osób !).

Pierwsze kompilatory powstały bardzo szybko, bo już w 1960 roku opracowane zostały przez firmy ADR (*Applied Data Research Inc*.), Remington Rand i RCA (*vide* "Zestawienie pierwszych kompilatorów języka COBOL").

COBOL 61 był rezultatem działania specjalnej grupy (*special task group*), której zadanie polegało na "zrewidowaniu" pierwszej wersji COBOLu. W 1962 roku opublikowano COBOL 61 Extended, do którego dodano, wzorując się na języku FACT, m.in. *Report Writer* i *Sort*. Rozpoczęto wówczas również prace na oprogramowaniem pamięci masowych o dostępie selektywnym (wyrywkowym). W 1966 roku opublikowano COBOL-65 zawierający właściwości operowania na zbiorach w tego rodzaju pamięci. Do roku 1967 jedynym posiadaczem kompilatora tej wersji była firma IBM.

<62>

Od 1963 roku rozpoczyna się standaryzacja COBOLu przez ASA

(*American Standards Association*). W 1960 roku powołano w ASA (przy ścisłej współpracy z *The Business and Equipment Manufacturers Association*) komitet standardów w dziedzinie komputerów i przetwarzania informacji Committee X3.

Komitet ten nie miał za zadanie opracować nowych wersji COBOLu, lecz tylko czuwać nad standaryzacją rezultatów działania komitetu PLC (Programming Language Committee) działającego w ramach CODASYLu. ASA (USASI, ANSI) Standard COBOL powinien być więc podzbiorem wersji CODASYL COBOL Specification, uznawanej przez użytkowników COBOLu jako jedyna "oficjalna" wersja tego języka.

Organizacja opisu COBOLu jest inna w wydawnictwach ANSI (USASI) niż CODASYLu. W pierwszym przypadku stosowany jest mianowicie tzw. modułowy opis (a nie rozdziałowy). Rozróżnia się 8 modułów: *Nucleus*, *Report Writer*, *Sort*, *Segmentation*, *Random Access*, *Sequential Access*, *Library*. Każdy moduł dzieli się na 2-3 poziomy zależnie od stopnia złożoności. ANSI COBOL 68 składa się z dwóch części: COBOLu właściwego i zestawu programów kontrolnych (*a set of audit routines*). Zadaniem zestawu kontrolnego jest kontrola dowolnej wersji COBOLu na zgodność z wersją standardową ANSI. Wersja ta posiada szereg nowości, np. deklaracja `SYNCHRONIZED` może być używana na poziomie rekordu, występuje instrukcja `WRITE .. AFTER .. POSITIONING`.

<63>

ZESTAWIENIE PIERWSZYCH KOMPILATORÓW JĘZYKA COBOL

Maszyna	Data wprowadzenia	Wersja	Uwagi
RCA 501	wrzesień 1960	COBOL 60	
ICT 1301		COBOL 60	publ. październik 1960
B – 5000		COBOL 61	Burroughs, wrzesień 196l
NCR 304 GE	grudzień 1961		
UNIVAC 490	wrzesień 1962		
UNIVAC 1105	wrzesień 1961	COBOL 61	HQ. AIR FORCE LOGIS COMMAND
H – 400	wrzesień 1962	COBOL 61	HONEYWELL + CSC
IBM 705-III	1961	COBOL 61 CODASYL	
IBM 705-II	luty 1962		
IBM 1401	marzec 1962		
IBM 1410	styczeń 1962		wykorzyst. IOCS
MOBIDIC	lipiec 1962		Sylvania Electric
GAMMA 30	lipiec 1962	COBOL RCA 301	RCA
ICT 1500	1962	RAPIDWRITE	na bazie COBOLu 61
KDF 9	grudzień 1963	COBOL 61	English Electric

<64>

ZESTAWIENIE PIERWSZYCH JĘZYKÓW DO PRZETWARZANIA DANYCH [X9] (poza językiem COBOL)

Język	Data wprowadzenia	Maszyna	Uwagi
GECOM firma GE	wrzesień 1961	GE-225	dopuszcza TABSOL, COBOL 61, niektóre cechy ALGOLu i FRINGE, 12 przebiegów lub mniej
FACT HONEYWELL + Comp. Sciences	grudzień 1961	H-800	8 przebiegów, 40 oper. min. sort., redagowanie
FLOWMATIC Sperry Rand	1956	UNIVAC I UNIVAC II	
UNICODE Sperry Rand	listopad 1959	UNIVAC 1103A	automatyczna segmentacja
JOVIAL	koniec 1961	IBM 7090 Philco 2000 CDC 1604 Q 7	rozwinięcie ALGOLu
NEBULA Ferranti	koniec 1962	ORION ATLAS	
ADAPT Comp. Sciences	październik 1961		
CODEL		ICT 1301	wycofany na rzecz COBOLu

<65>

Ponieważ omawiamy genezę COBOLu, warto wymienić konkretne zapożyczenia tego języka oraz oddziaływania na inne języki. Tak na przykład, instrukcja warunkowa *IF .. THEN* oraz *PICTURE* pochodzą z COMMERCIAL TRANSLATOR (pierwsza specyfikacja tego języka – styczeń 1958), stanowiąc obecnie typowe zwroty COBOLu. Istotnym źródłem dla zespołu roboczego COBOLu (Short Range Committee) był język FLOW-MATIC (pierwsza wersja – 1957), zawierający takie wyrażenia jak: *DIVIDE .. BY .. GIVING*, *MOVE .. TO .. , IF .. GO TO ..* oraz strukturę rozdziałową (rozdzielanie opisu danych od procedur).

Próbą rozszerzenia COBOLu był język IDS (*Integrated Data Store*) opracowany w General Electric. Uwzględniał on w szczególności właściwości "łańcuchowego" (*chaining*) adresowania w pamięciach dyskowych. Do opisu danych wprowadzone zostało tzw. "pole łącznika" oraz takie deklaracje jaki *RETRIEVAL VIA CALC CHAIN*, *PLACE NEAR n-d CHAIN*, *PAGE-RANGE*, instrukcje: *STORE* (powoduje

utworzenie łańcucha), *RETRIEVE*, *MODIFY*, *DELETE*. Dla maszyn PDP-81, PDP-8L firma Digital Equipment Corp. opracowała język DIBOL (*Digital equipment Business-Oriented Language*), klasyfikowany jako cobolopodobny (*cobol-like*) i składający się z trzech komponentów: procesora, systemu operowania danymi (do operowania na zbiorach bez dodatkowego programowania przez programistę piszącego program użytkowy) i monitora, który łączy poprzednie części w jedną całość i umożliwia użytkownikowi łatwe posługiwanie się językiem.

W listopadzie 1963 roku organizacja ECMA (*European Computer Manufacturers Association*) wydała propozycje COBOLu dla małych komputerów; określone jako COMPACT COBOL. Wersja ta zawiera mniej więcej połowę zwrotów pełnego języka. Kompilator

<66>

dla niej opracowała firma ICT, poprzedzając tym prace nad właściwym COBOLem.

Pod względem zakresu COMPACT COBOL odpowiada COBOLowi B Honeywell (potrzebującemu jedynie 8K znaków pamięci operacyjnej.

W 1962 roku firma ICT opracowała dla maszyny ICT 1500 system RAPIDWRITE, przedstawiający sobą sposób "szybkiego zapisu" programu zewnętrznego z użyciem najistotniejszych (niezbędnych) członów wyrażeń COBOLu. Program pisze się na specjalnych arkuszach z nadrukami. Zadaniem translatora RAPIDWRITE jest przetłumaczenie zdań uproszczonych na zdania pełne COBOLu.

Podobny do COBOLu jest język ADAPT, opracowany w 1961 roku przez Computer Sciences dla IBM 1401.

Koncepcją wykorzystującą dorobki różnych języków jest system GECOM (GEneral COMpiler) opracowany przez General Electric dla maszyny GE 225 i wdrożony w 1961 roku. System ten oparty został o języki: COBOL, ALGOL, FRINGE (do sortowania, wydruków, aktualizacji zbiorów) i TABSOL (do tłumaczenia tablic decyzyjnych).

4. Geneza języka PL/I

PL/I (w zasadzie skrót nieprzetłumaczalny, czasem interpretowany [H1] jako *Programming Language No 1*) powstał poniekąd przypadkowo. Otóż w związku z projektowaniem serii 360 firma IBM postanowiła ulepszyć FORTRAN, który posiadał znaczne niedostatki w operowaniu znakami i danymi alfanumerycznymi oraz nie bardzo nadawał się do współdziałania z nowoczesnym systemem operacyjnym maszyn wieloprogramowych.

We wrześniu 1963 roku organizacja użytkowników FORTRANu

<67>

nosząca nazwę SHARE i firma IBM powołały, zgodnie z projektem SHARE FORTRAN ADVENCED LANGUAGE DEVELOPMENT COMMITTEE. Wkrótce okazało się, że produkt pracy zespołu wbrew założeniom nie będzie kompatibilny (zamienny) z FORTRANemi należy opracować odrębny nowy uniwersalny język programowania, który nazwano NPL (*New Programming Language*). Nazwa ta nie utrzymała się, ponieważ National Physical Laboratory (NPL) zgłosiła protest przeciwko używaniu skrótu NPL i nazwę języka zmieniono na PL/I.

W trakcie opracowywania NPL przestudiowano zalety takich języków jak ALGOL, COBOL i JOVIAL (oprócz FORTRANu). Po raz pierwszy zaprezentowano opis języka w marcu 1964 roku, zaś w kwietniu, czerwcu i grudniu podano następne wersje. W 1965 roku powstał pierwszy oficjalny podręcznik programowania.

Do roku 1966 powstało 6 wersji języka NPL, PL/I: 0, 1, 2, 3, 4. Dopiero po wszechstronnej krytycznej dyskusji firma IBM zdecydowała się na opracowanie kilku translatorów, Pierwszy translator PL/I został opracowany przez Zakład Badawczy (Laboratory) IBM w Hursley (Anglia) i ukończono go w sierpniu 1966 roku. Zakład ten oraz Vienna Laboratory posiadają duże zasługi również w pracach nad zdefiniowaniem języka.

Pierwsze wersje języka PL/I wzbudziły duże zainteresowanie wśród producentów i już pod koniec 1965 roku mały podzbiór PL/I zwany NICOL I został wdrożony przez Massachusetts Computer Associates na maszynie IBM 7094, BELL Laboratories opracował wersje APL, zaś Massachusetts Institute of Technology (MIT) stworzył MULTICS. Niemniej jednak do 1959 roku język PL/I wdrożono jedynie na małej liczbie maszyn dużej wielkości i był on dostępny w zasadzie jedynie dla członków organizacji SHARE i GUIDE [X12].

<68>

VI. Zamienność języków i programów

Zamienność języków i programów (napisanych w różnych wersjach tego samego języka) stanowi kluczowy problem rozwoju programowania. Chodzi oto, by nie dopuścić do sytuacji, iż będzie tyle języków, ile jest (i było) maszyn. Pozytywne rozwiązanie problemu umożliwi wykorzystanie dotychczasowego dorobku programowania i kwalifikacji posiadanej kadry kilkuset tysięcy (w skali światowej) programistów.

Stąd usiłowania standaryzacji języków prowadzone przez organizacje międzynarodowe (ISO, ECHA) i amerykańskie (CODASYL, USASI) oraz wysiłki niektórych rządów (np. USA w przypadku COBOLu). Działalność ta, z takich czy innych powodów, nie jest jeszcze należycie uwzględniana przez producentów, posiadających swoje własne ambicje

i dla dobra walki konkurencyjnej (a niekiedy i postępu), stale wprowadzających innowacje, nie zawsze do końca przemyślane. Inaczej nie mielibyśmy paru tysięcy języków wyższego rzędu, z których wiele stawia sobie takie same zadania, częściowo dubluje składnię oraz ... błędy poprzedników. Ponadto "oficjalne" (teoretyczne) wersja niektórych języków tak abstrahują od technicznej i organizacyjnej strony przetwarzania (np. ALGOL w zakresie wejścia-wyjścia), że ich wdrożenie jest możliwe jedynie poprzez dopracowanie przez producentów i instytucje software'owe.

Najpoważniejszym czynnikiem różnicującym wersje są różnice w wyposażaniu komputerów. W przypadku bardziej złożonych języków (jak COBOL, PL/I), wymagających pojemnych pamięci operacyjnych, często następuje "obcinanie" pełnych wersji, przystosowując je do aktualnych możliwości sprzętu.

<69>

W warunkach dużej dynamiki rozwoju konstrukcji i programowania standaryzacja jest bardzo utrudniona. Można powiedzieć, że jeśli stosuje się ją przedwcześnie, to "konserwuje" ona elementy błędne i mało rozwojowe, hamując tym samym dalszy rozwój dziedziny.

Na pewno zamienność absolutna jest nie do osiągnięcia. Potrzebny jest kompromis ukształtowany pod wpływem kosztu przedsięwzięcia, efektywności kompilatorów itp.

Wydaje się, że standaryzacja programowania powinna być poprzedzona standaryzacją pewnych podstawowych elementów hardware'owych jak np. standaryzacja taśm magnetycznych, rodzajów kodów znaków (łącznie z problemem starszeństwa) itp. Ponadto więcej uwagi należałoby poświęcić rozwojowi technik reprogramowania z maszyny na maszynę (emulacja, symulacja itp.).

1. **Koncepcja uniwersalnego języka pośredniego**

Wzrost liczby języków programowania oraz typów maszyn powoduje znaczne zwiększenie pracochłonności opracowania translatorów. Jest ona proporcjonalna do iloczynu tych dwóch elementów (tj. liczby języków przez liczbę typów maszyn). Gdyby udało się we wszystkich maszynach zastosować jeden uniwersalny język pośredni, to liczba translatorów tego języka równałaby się liczbie maszyn (LM). Z kolei wymagane byłoby przetłumaczenie każdego języka użytkowego na uniwersalny język pośredni i do tego celu potrzeba byłoby tyle translatorów, ile jest języków (LJ). Ogólna liczba translatorów równa się więc sumie $LM + LJ$, podczas gdy w pierwszym przypadku występował iloczyn. Różnica jest więc oczywista (nie tylko arytmetycznie), nawet jeśli uwzględnimy, że to liczbowe ujęcie jest niedokładne,

<70>

ponieważ nie uwzględnia różnic w pracochłonności opracowania translatora języka uniwersalnego i translatora języka specjalistycznego (w iloczynie oba argumenty przedstawiają ilości języków specjalistycznych, podczas gdy w sumie argumenty są mieszane).

Koncepcja języka uniwersalnego UNCOL (*Universal Computer Oriented Language*) pojawiła się pod koniec lat pięćdziesiątych (lata 1958-1961) w artykułach Convey'a M.E., Steela T.B., Stronga J. Niestety, języka takiego nie opracowano, a na przeszkodzie stanął zakres pracy, jaki należało wykonać: uzgodnienie, jakie elementy mają być uniwersalne, wybór poziomu programowania oraz aktualizacja pierwszych wersji w związku ze zmianami konstrukcji maszyn.

Koncepcja języka pośredniego nie została jednak całkowicie zarzucona. W charakterze takiego języka w ramach poszczególnych firm występuje zwykle język symboliczny (np. Język PLAN dla serii 1900). Niekiedy językiem pośrednim jest język wyższego rzędu, np. FORTRAN stosowany jest jako język pośredni dla APT. Programy APT są za pomocą specjalnego podzbioru APT redagowane wstępnie (zmiana składni) na formę akceptowaną przez kompilator FORTRANU [G3].

2. **Emulacja**

Problemy zamienności języków i programów są równocześnie problemami zamienności komputerów. Stąd też są rozwiązywane m. in. metodą opartą zarówno o środki programowe jaki hardware'owe, zwaną emulacją. Środkiem programowym jest specjalny program, zaś hardware'owym – specjalna pamięci dodatkowe rejestry. W "emulator" wyposaża się zwykle maszynę nowszą (zakupioną na

<71>

miejsce poprzedniej), po to, by mogła akceptować dotychczasowe programy opracowane dla maszyny wycofywanej (starszej). Emulacja nie zapewnia idealnej zamienności i doprowadza do zamienności kody operacji itp., a więc w przypadku zastosowania różnych urządzeń peryferyjnych i systemów operacyjnych pewne przepracowanie programów będzie konieczne.

Oto "zamienne" maszyny firmy IBM w stosunku do serii 360 (wg [X13]):

1620	model 30
1401, 1440, 1460, 1410, 7010	40

7070/7074	50
705/7080 709/7090	65

Hardware'owym elementem emulacji dla serii 360 jest pamięć pasywna ROM – *Read Only Memory* (zwana też ROS – *Read Only Storage*), zawierająca mikroprogramy. W miejsce klasycznego wykonywania instrukcji, pamięć ta powoduje otwieranie i zamykanie szeregu elektronicznych "bramek" w systemie obiegu impulsów. Elementem software'owym jest program emulatora, umieszczony w pamięci ferrytowej, służący do wykonywania programu obcej maszyny (również znajdującego się w tej pamięci) w konwencji serii 360. "Obcy" program znajduje się pod kontrolą ROM dopóki nie zostanie napotkana instrukcja, która nie może być hardware'owo interpretowana. Wtedy sterowanie przekazywane jest do programu emulatora.

Pod względem funkcjonalnym emulacja podobna jest do symulacji. Wykonywana jest jednak efektywniej dzięki mikroprogramom, które wykonują funkcje: interpretacji instrukcji, dekodowania adresu, wybierania zawartości adresu

<72>

i wykonania wskazanej operacji. Czynności te są najbardziej pracochłonne i pochłaniają dużo czasu przy symulacji.

Wykonanie programu techniką emulacji wymaga większej pamięci operacyjnej (do przechowywania specjalnego programu), zaś efektywność programu adaptowanego na serię 360 powinna być nie mniejsza, niż na maszynie pierwotnej (mimo, iż możliwości serii 360 nie są w pełni wykorzystywane). Pamięć ROM zbudowana być może z różnych elementów. Model 30 serii 360 w charakterze ROM posiada specjalne miedziane karty perforowane. W innych maszynach (np. firmy Burroughs) ROM zbudowana jest na układach scalonych.

3. Symulacja

Słownik IFIP [V1] pod hasłem A2 podaje następującą definicję symulacji: "Symulacja. Reprezentowanie pewnych właściwości zachowania się jednego systemu poprzez działanie innego systemu". Symulacja działa podobnie jak emulacja lecz bez środków hardware'owych. W pamięci przechowywany jest program obcy i program symulator. Każdy rozkaz może być interpretowany i wykonywany, iw tym przypadku nie trzeba w pamięci przechowywać całego programu obcego. Firma ICL opracowała następujące symulatory (dla serii 1900):

>– Elliott 803 Simulator (*xME4* , *xME8*). Przeznaczony jest do

symulacji emc Elliott 803 z 4K PAO (lub 8K). Na maszynie 1905 programy Elliott 803 są wykonywane przeciętnie dwukrotnie szybciej (przy czym zapotrzebowanie na PAO wynosi 9600 słów lub 17792).

– Pegasus Simulator

xMP4 dla	4K Pegasus i 12K 1900
xMP7	7K Pegasus i 18K 1900

<73>

Symulator ten staje się niepraktyczny w przypadku używania ręcznych kluczy (przełączników na pulpicie) przez programy maszyny Pegasus.

4. Translacja z języka na język

Jest to najmniej efektywny sposób przerabiania programu s języka na język, ponieważ polega na "mechanicznym" przekształcaniu instrukcji, bez uwzględniania specyfiki nowej maszyny. Wymaga też większej pamięci, ponieważ najpierw tłumaczony jest cały program, a dopiero później wykonywany.

Jako przykłady translatorów wymienić można następujące prace firmy ICL:

– ATLAS ALGOL na 1900 ALGOL (*xAC2*)

– MPL (1300) na PLAN 1900 (*xJP3*)

W xJP3 rozkład danych na bębnie jest "symulowany" w pamięci ferrytowej. Nie podlega tłumaczaniu E funkcja oraz zmodyfikowane instrukcje I. Zakłada się, że program pisany w MPL jest bezbłędny (nie ma kontroli diagnostycznej).

Z publikacji [O1] znana jest koncepcja systemu XACT (*X Automatic Code Translation*, przy czym *X* oznacza dowolną maszynę), opracowanego od 1961 roku przez Celestron Associates, Inc. aa bazie pary maszyn 7090 – 1604. Maszyna 7090 jest maszyną źródłową (*source*), zaś 1604 – docelową (*target*). Problem polega na tym, by tak przekształcić programy 7090, by mogły być realizowano na 1604. Ocenia się, że opracowanie translatora dla tej pary wymaga 10-15 osobolat pracy programistów. Ciekawostką jest to, że translacja przebiega (przynajmniej częściowo) nie ma maszynie docelowej, lecz aa maszynie źródłowej (!), do której wprowadza się poza programem również opis

<74>

danych (liczb). W pierwszej fazie kompilacji tworzony jest maszynowo

niezorientowany opis funkcji do wykonania, zaś w drugiej powstaje programów kodzie maszyny docelowej.

5. Uwagi ogólne

Można wyróżnić następujące typy tłumaczeń z języka na języki

1. języka maszynowego jednej maszyny na maszynowy język innej maszyny (szczególne trudności występują tutaj m.in. przy tłumaczeniu programów wejścia-wyjścia),

2. języka maszynowego na język zewnętrzny (mamy tutaj do czynienia z dekompilacją, czyli procesem odwrotnym do kompilacji)

3. języka zewnętrznego na wewnętrzny,

4. języka zewnętrznego na język zewnętrzny,

5. translatora na translator (być może ten rodzaj tłumaczenia ma większe perspektywy rozwojowe, niż tłumaczenie konkretnych programów użytkowych).

Automatyczne tłumaczenie wymaga wstępnej selekcji materiału, odrzucającej zwroty nieprzetłumaczalne i błędne. Dlatego też systemy tłumaczące powinny posiadać organizację umożliwiającą programiście ingerencję w proces tłumaczenia oraz zapewniającą sygnalizację elementów błędnych lub do ręcznej konwersji.

Istnieje szereg sposobów postępowania w przypadku napotkania trudności w tłumaczeniu. W systemie tłumaczenia z języka symbolicznego IBM 7090 na język maszynowy IBM 7040, nieprzetłumaczalnym bezpośrednio zwrotów przydziela się "fikcyjne" makrorozkazy (ekstrakody), które są rozszyfrowywane dopiero w następnych przebiegach. Oto inne rodzaje trudności [G4].

a) cały program lub bloki programów są całkowicie nieprzetłumaczalne – ma to miejsce wtedy, gdy konstrukcja maszyny

<75>

Jest częścią algorytmu (np. program testowania pamięci operacyjnej maszyny CDC 1604),

b) samomodyfikacja programu (dynamiczna struktura programu),

c) wyrażenia idiomatyczne (gdy niektóre rozkazy lub ich sekwencje mają sens jedynie wtedy, gdy rozpatrywane są na tle całego programu),

d) różnice konstrukcyjne pomiędzy maszynami (np. słowo 24-bitowe i 36-bitowe).

6. **Zamienność programów pisanych w COBOLu**

Wersje oficjalne (np. ogłoszone przez CODASYL) COBOLu są co prawda maszynowo niezależne, lecz stanowią one dla twórców kompilatorów jedynie bazę koncepcyjną, do której wprowadzają własne pomysły, utrudniające zamienną eksploatację programów. Pomysły te stanowią odbicie specyfiki hardware'owej lub też są po prostu innowacją formalną w składni języka (np. w wersji COBOLu na maszynę ZAM41 opracowanej przez Instytut Maszyn Matematycznych w Warszawie).

Rozważając problemy zamienności programów, trzeba wspomnieć o przypadkach obiektywnej niezamienność, np. gdy program napisany w wersji dyskowej (z zastosowaniem indeksowania lub randomizacji) chcemy eksploatować na maszynie wyposażonej tylko w taśmy magnetyczne.

Już od samego początku podejmowano starania, by COBOLowi zapewnić wysoki stopień zamienności. Tak np. w grudniu 1960 roku demonstrowano zamienność programów COBOLu pomiędzy maszynami RCA 501 i UNIVAC II [B10]. Później nadzór nad składnią tego języka przejęły takie organizacje międzynarodowe (poza

<76>

CODASYLem), jak ASA/USASI, ANSl/ISO, ECMA. Zakres wersji kompilatora Cobolu i jego efektywność zależą przede wszystkim od wielkości dostępnej (po odjęciu obszaru dla systemu operacyjnego itp.) pamięci operacyjnej. W celu sklasyfikowania poszczególnych wersji firmy IBM i Honeywell wprowadziły pojęcie poziomu (*level*) języka: Honeywell – B, D, H, I, F, zaś IBM – E, F. Każdy poziom języka różni się zakresem słownika dopuszczalnych zwrotów. Firma ICL (seria 1900) usiłowała natomiast stosować jedną wersję języka zmniejszając efektywność kompilacji przy mniej dogodnych konfiguracjach (użycie wielokrotnego wzywania segmentów wymiennych).

Dla programisty najważniejszym problemem jest niewątpliwie zakres czynności, jakie ma wykonać, chcąc zaadoptować program COBOLu pisany dla innej maszyny. Brakuje tego rodzaju informacji w publikacjach krajowych i obcych. Poniżej podamy pewne zasady oparte o własne doświadczenia w programowaniu na maszyny ICL 1900, ICL System 4-50 oraz ZAM-41.

1. porównanie konfiguracji maszyn (w szczególności pojemności pamięci, rodzajów urządzeń wejścia-wyjścia i pamięci masowej),

2. porównanie listy wyrażeń obu kompilatorów oraz sprawdzenie zakresów działania takich samych zwrotów (m.in. dotyczy to deklaracji *COMPUTATIONAL*, *DISPLAY*-n ściśle związanych z postacią liczb w maszynach – postać binarna, pojedyncza lub podwójna precyzja, stały lub zmienny przecinek, obliczenia w angielskim dziesiętnym systemie

walutowym itp. W COBOLu i Honeywell I deklaracje *COMP*, *COMP-1* posiadają identyczne znaczenie),

<77>

3. sprawdzenie, czy program jest segmentowany i ile miejsca zajmuje w PAO w dotychczasowej posegmentowanej postaci,

4. sprawdzenie, czy program posiada wstawki napisane w innych językach (w języku symbolicznym lub wyższego rzędu np. w FORTRANie lub języku symbolicznym dla COBOLu Honeywell). Jeśli takie wstawki występują, to należy zapewnić odpowiednik wstawki w "macierzystym" języku wstawek i wprowadzić odpowiedni aparat formalny związany z użyciem instrukcji *ENTER*, *CALL* itp. Można oczywiście zamiast wstawki napisać fragment programu w języku COBOL.

5. porównanie sposobów wejścia danych; należy ustalić, czy występuje zapis pozycyjny i niepozycyjny. Jeśli stosowany jest zapis niepozycyjny, to należy sprawdzić, jakie znaki są stosowane w charakterze standardowych ograniczników.

6. porównanie treści metryk standardowych (w szczególności dotyczy to zbiorów na dyskach i taśmach magnetycznych),

7. porównanie parametrów drukarek wierszowych (długość wiersza, znaki specjalne, funkcje poszczególnych kanałów taśmy sterującej itp.).

Jako przykład firmowych odrębności można podać zwroty występujące w COBOLu Honeywell:

– *STOP JOB* -- zakończenie ciągu zadań obok *STOP RUN* (np. w celu przejścia do podprogramu),

– *CANCEL* -- zwolnienie obszaru pamięci operacyjnej, zajętego przez podprogram,

– *LOAD* -- załadowanie podprogramu do pamięci operacyjnej.

Można powiedzieć, że ani jednego programu (z wyjątkiem trywialnych) COBOLu nie można mechanicznie przenieść na inny typ maszyny bez dokonania pewnych zmian.

<78>

Najbardziej zmienną częścią COBOLu jest ENVIRONMENT DIVISION, a więc rozdział opisujący konfigurację maszyny, najmniej -- co nie oznacza, że w ogóle nie ulega zmianom – PROCEDURE DIVISION. DATA DIVISION jest z omawianego punktu widzenia czymś pośrednim:

mogą wystąpić różnice w opisie metryk zbiorów w klauzulach typów danych w długości wiersza na tabulogramie itp.

Jeśli chodzi o część proceduralną to nie we wszystkich wersjach występują bardziej złożone warianty instrukcji *PERFORM* (z *AFTER*), zwrot *DEPENDING ON* (w połączeniu z *GO TO*), *MOVE CORRESPONDING* itp. Pewien kłopot sprawić może uzyskanie takiej samej dokładności wyniku przy operacjach arytmetycznych. Do innych rezultatów doprowadzić mogą operacje porównywania pól znaków alfanumerycznych w sytuacji, gdy porównywane maszyny posiadają inną sekwencję starszeństwa znaków (np. w niektórych maszynach litery są "większe" od cyfr w innych – mniejsze). Odrębne zagadnienie stanowi wpływ systemu operacyjnego maszyny, na którą opracowano program, stosowanie pakietów REPORT WRITER, procedur diagnostycznych typu *READY TRACE*, *RESET TRACE*, *EXHIBIT* ...

Pomocą w ocenie zamienności kompilatorów mogą byś specjalistyczne generatory programów np. CCVS (*COBOL Compiler Validation System*) opracowany przez US Air Force [H7] dostarczające odpowiedzi na najistotniejsze pytania:

a) czy kompilator spełnia wymogi poziomu, do którego został zaliczony, tzn. czy rzeczywiście wykonywane są wszystkie funkcje przewidziane przez standard poziomu,

b) czy program wewnętrzny otrzymany po kompilacji dostarcza wyniki odpowiadające określonym standardom. Analiza zamienności różnych kompilatorów stanowi interesujący

<79>

i obszerny problem kwalifikujący się do odrębnego opracowania. Ze względu na ograniczoną objętość niniejszej pracy, nie możemy poświęcić temu więcej miejsca.

Przykłady różnic pomiędzy wersjami języka COBOL

Lp.	Element	ICL seria 1900	ICL System 4-50	IMM ZAM 41	Uwagi
1.	Nazwy danych i nazwy paragrafów	1-szy znak musi być literą	w nazwie danych musi być co najmniej jednak litera, ale nie musi być to 1-szy znak nazwa paragrafu może być liczbą	1-szy znak musi być literą	
2.	Nazwy urządzeń np. czytnika kart	CARD-READER	UNIT-RECORD C-READER	INPUT (nr)	IBM 360: UNIT--RECORD 1402R
3.	Deklaracja np. COMPUTATIONAL	liczba dziesiętna kodowana dwójkowo (*binary decimal*) [1]	postać binarna	--	

4.	COMPUTA-TIONAL-1	liczba binarna	liczba zmienno-przecinkowa	--	
	FIXED	--	--	liczba całkowita binarna	
	FLOAT	--	--	liczba zmienno-przecinkowa w postaci binarnej	

[1] *Nie jest to typowa postać liczby dziesiętnej kodowanej dwójkowo, ponieważ co prawda jeden znak zajmuje 4 bity, lecz dwa znaki -- 7 bitów, trzy znaki -- 10 bitów.*

<80>

7. **Zamienność programów pisanych w Fortranie**

Podstawową wskazówką zamienności jest rodzaj podstawowej wersji : FORTRAN, FORTRAN II, FORTRAN IV. Ponadto w ramach tych samych wersji mogą występować znaczne różnice, np. kompilatory FORTRANu dla IBM 1620, IBM 650 nie zawierały deklaracji *FORMAT* i nie dawały możliwości wezwania podprogramów.

Następujące zwroty FORTRANu IV nie występowały w FORTRANIE II [G2]: *DATA*, *BLOCK DATA*, *NAMELIST*, logiczne *IF*, numery zdań w charakterze argumentów podprogramów, wielokrotne wejścia do podprogramów.

Oprócz tego, istnieją następujące różnice w instrukcjach wejścia-wyjścia.

FORTRAN II	FORTRAN IV
READ INPUT TAPE i,j, parametry	*READ(i,j)* parametry
WRITE OUTPUT TAPE i,j, parametry	*WRITE(i,j)* parametry
READ TAPE j, parametry	*READ(j)* parametry
READ DRUM i,j, parametry	*READ(k)* parametry
WRITE DRUM i,j, parametry	*WRITE(k)* parametry

Opracowany został translator SIFT (Share Internal FORTRAN Translator) do modyfikacji programów FORTRANu II na programy FORTRANu IV.

<81>

8. **Zamienność programów pisanych w ALGOLU**

Kłopoty z zamiennością programów dotyczą przede wszystkim procedur wejść-wyjść,

opisywanych nie przez wersje teoretyczne definicji języka, ale przez translatory (kompilatory). Pogląd ten zilustrujemy w poniższym zestawieniu [1] .

[1] *Przyjęto oznaczenie* `n, m ... n1, n2, ... nn` *-- wartości liczbowe,* `nd` *-- nazwa danych. [przyp.2020]*

Lp.	Procedury	ICL seria 1900	ICL System 4-50	ODRA 1204	GIER ALGOL	Ural 2
1.	Przydział urządzeń	select input(n) select output(n)	OPEN(n) SET(n,m)	set input(n) set output(n)		
2.	Zwolnienie urządzeń	free input free output	CLOSE(n)			
3.	Operacje czytania, pisania itd.	print(E,n,n) output(E) write boolean(E) write text(...) copy text(...) N := read	WRITE(n,FORMAT...nd) nd := READ(n)	print(n1...nn) ininteger inreal inchar print(n1...nn) outchar line(n) read(n1...nn)	write writetext write er write char output outchar outsum input inone inchar setchar lyn typein typechar	writetext outtext outer writeer outchar outclear input inone typein

<82>

VII. Rozwój programowania a rozwój konstrukcji maszyn

Programowanie i konstrukcje stanowią dwa wzajemnie warunkujące się elementy składowe tego, co nazywamy komputerem. Rozwój tych elementów nie odbywa się jednak w idealnej harmonii. W początkach rozwoju komputerów, kiedy dopiero wykrywano ich możliwości, sprzężenia zwrotne pomiędzy tymi elementami było szybsze. Rozwiązania przebiegały w stosunkowo małej skali (niewiele koncepcji mało indywidualnych konstrukcji, brak walki konkurencyjnej dużych firm) stąd nietrudno było o koordynację i szybkie rezultaty. W miarę rozkręcania wyścigu o rekordy szybkości działania coraz mniej uwagi udzielano sprawom programowania. Ponadto opóźnienie tego ostatniego wynikało z naturalnej kolejności opracowania komputera (mimo pewnej równoległości prac, software opracowywuje się w zasadzie dla gotowych konstrukcji). Ponieważ cykl opracowania software'u jest o wiele dłuższy od konstrukcyjnego przygotowania, często nie nadążano po prostu z dopracowaniem go przed zejściem nowego modelu komputera z. taśmy produkcyjnej. Wg I. L. Auerbacha [A2] system operacyjny IBM 360 jest opóźniony o 5 lat w stosunku do rozwiązań konstrukcyjnych.

W pewnych przypadkach konstrukcje są wyprzedzane przez pomysły software'owe (np. pamięci asocjacyjne).

Tworzy się maszyny IV generacji o wysokim stopniu modularności obejmującej oprócz dotychczas stosowanych procesorów wejścia-wyjścia również niezależne procesory centralne i niezależne (o równoczesnym dostępie) moduły pamięci operacyjnej. Stwarza to nowe problemy zarówno hardware'owe jaki software'owe.

<83>

A. Spróbujmy prześledzić rozwój urządzeń pamięciowych, stanowiących główną przesłankę rozwoju programowania (służą do przechowywania programów użytkowych, translatorów, systemu operacyjnego-dyrygenta, biblioteki programów itp.).

A.1 Początkowe do przechowywania informacji używane były przerzutniki (np. w ENIACu – przerzutniki lampowe – jako pamięć operacyjna). Następnie do połowy lat pięćdziesiątych stosowano pamięci dynamiczne na rtęciowych (po raz pierwszy – 1949 EDSAC) oraz niklowych (od 195l) liniach opóźniających. Można wspomnieć również o pamięciach elektrostatycznych na lampach kineskopowych (po raz pierwszy budowanych przez Prof. Williama w Manchester University oraz w MIT). Pamięci dynamiczne i elektrostatyczne, aczkolwiek niedoskonałe (wolne i małopojemne – 512, 1024 słów) stanowiły w porównaniu z ENIACem duży krok naprzód, ponieważ umożliwiały przechowywanie programów w pamięci. Wystąpiła więc możliwość stosowania instrukcji skoków oraz wzywania podprogramów.

A.2 Jeszcze przed pamięciami na liniach opóźniających, bo mniej więcej od 1947 roku używano bębna magnetycznego, niemniej jednak z powodu dużego czasu dostępu nie bardzo nadawał się na pamięć operacyjną. Oto pierwsze zastosowania bębna (koniec lat 40-tych).

– SEC (*Simple Electronic Computer*) zbudowany na uniwersytecie londyńskim,

– ERA 1101, zbudowana w 1950 roku przez Engineering Research Associates,

– MARK III, MARK IV, zbudowane w Harvard University w latach 40-tych.

<84>

Pierwszą maszyną używającą bębna w charakterze pamięci pomocniczej była maszyna WHIRLWIND I, której budowę zakończono w 1951 roku w MIT. Pamięć bębnową posiadała maszyna IBM 650, będąca najpopularniejszą maszyną na świecie w latach 50-tych. Maszyna ta posiadała środki programowe (system SOAP) do optymalizacji wybierania danych z bębna.

A.3 Głównym rodzajem pamięci operacyjnej, niezastąpionym od lat kilkunastu, jest pamięć ferrytowa. Teoretyczne artykuły na temat tego typu pamięci zaczęły pojawiać się dopiero na początku lat pięćdziesiątych. Niemniej jednak Już w 1951 roku firma Jacob

instrument Company wprowadziła tę pamięć do maszyny Jain Comp D-1 [M3]. Poważne prace badawcze nad pamięciami ferrytowymi prowadzone były w MIT od mniej więcej 1953 roku.

Firma IBM użyła po raz pierwszy rdzeni ferrytowych w maszynach 704 i 705 (około 1955 r.). W marcu 1956 roku zastosowała je firma Remington Rand w maszynie UNIVAC 1103A.

Wprowadzenie pamięci ferrytowych zapewniło maszynom dużą szybkość (mniej więcej tysiąckrotnie wyższą niż w przypadku bębna), umożliwiło zastosowanie podziału czasu (a więc i pracy wieloprogramowej). Dzięki dużym pojemnościom (od kilku tysięcy do kilku milionów słów) zaistniała możliwość użycia złożonych języków programowania i systemów operacyjnych. Ze względu na ciągłe wzbogacanie translatorów (m.in. o powiązania z systemem operacyjnym) uważa się, że obecny "standard" 32K, w maszynach IV generacji zostanie podwyższony do 64K.

Pamięci ferrytowe budowane mogą być "wyrazowo" lub "znakowo" względnie poskładać mogą konstrukcję mieszaną pozwalającą na bezpośrednie adresowanie zarówno znaków (lub byte'ów) jak i

<85>

słów (komórek). Do niedawna stosowano najczęściej jednostki pamięciowe 24-bitowe i 8 (lub 6)-bitowe. Ostatnio obserwujemy tendencje pośrednie, a mianowicie wprowadzanie pamięci o jednostce adresowania l6-bitowe. Jest to kompromis pomiędzy dwoma sprzecznymi walorami. Krótsze jednostki adresowania umożliwiają sprawniejsze manipulacje danymi (co jest szczególnie przydatne przy tłumaczeniach) oraz lepsze wykorzystanie pamięci (mniej pustych bitów). Pamięci "wyrazowe" natomiast zapewniają wyższe szybkości działania dzięki równoległemu przesyłaniu większej liczby bitów.

Podkreślaliśmy znaczenie pojemnej pamięci operacyjnej. Istnieją środki hardware'owe-programowe, określane jako tzw. stronicowanie [1] (*paging*), które umożliwiają programiście traktowanie pamięci, jakby była nieskończona. Na przykład, stronicowanie w maszynie IRIS 80 pozwala na pisanie programów o wielkości 16 milionów słów przy maksymalnej fizycznej wielkości pamięci 1 ml słów. Nie chodzi tutaj o klasyczną segmentację programów. Programista posługuje się teoretycznymi obszarami pamięci, zwanymi "stronami", podczas gdy grupa danych, znajdująca się fizycznie w pamięci, nazywana jest "blokiem" (niekiedy stosuje się odwrotne przyporządkowanie pojęć strona i blok). Problem polega na tym, by stronie przyporządkować odpowiedni blok, tj. dokonać konwersji adresu strony na adres bloku. Jeśli "strony" nie ma aktualnie w pamięci, to wtedy następuje automatyczne jej przesłanie bez udziału programisty. Do przechowywania informacji o aktualnie używanych stronach służą specjalne rejestry (w ICL 1906A jest ich 16), które wszystkie mogą być równocześnie angażowane. Jeśli żądany adres strony nie występuje w żadnym z nich, wtedy szuka się jej odpowiednika

<86>

za pomocą specjalnej tabeli (*special-look-up-table*).

[1] *Używa się też terminu "paginowanie" (Przyp.Red.).*

Duże nadzieje wiąże się z pamięciami budowanych na układach, scalonych. W

szczególności dotyczy to tzw. pamięci notatnikowych (*scratch-pad memory*). Czas dostępu do takiej pamięci jest minimalny: w maszynie SDS Sigma 7 – 150 ns (z roku 1966), w maszynie IRIS-80 – 60 ns (dla rejestru 16-bitowego).

A.4 Podstawowe znaczenie dla przetwarzania danych ekonomiczno-administracyjnych posiadają masowe pamięci pomocnicze. Za datę pierwszego użycia taśmy magnetycznej zwykło się podawać rok 1954, kiedy to zastosowano ją w maszynie UNIVAC I. Niemniej jednak już w 1949 roku (!) została ona wprowadzona do maszyny BINAC, zbudowanej tak, jak UNIVAC I przez Eckerta i Mauchly'ego [M3].

W maju 1955 roku firma IBM poinformowała o stworzeniu dysku magnetycznego (model 305). Pojemność dysku wynosiła 5-6 milionów znaków, zaś czas dostępu był dość znaczny (200-600 msek). W 1957 roku wprowadzono [K2] pamięć karuzelową (rodzaj pamięci taśmowej), zaś w 1961 roku zastosowano karty magnetyczne.

Prace nad pamięciami masowymi zmierzały w kierunku zwiększenia pojemności, zmniejszenia czasu dostępu i obniżenia kosztu.

Pojawienie się pamięci masowych postawiło przed programistami nowe zadania. W przypadku pamięci taśmowych opracowano szereg algorytmów sortowania (np. wg rozkładu Fibbonaci'ego, metodą kaskadową i polifazową) oraz ustalono, tzw. standardowe metryki zbiorów. Pamięci dyskowe okazały się trudne do opanowania od strony programowej ze względu na różnorodność sposobów przechowywania i wybierania informacji (sposoby te podajemy w terminologii angielskiej w celu uniknięcia nieporozumień:

<87>

serial, *sequential*, *self-indexing*, *partial indexing*, *full indexing*, *randomizing*).

Eksploatacja pamięci dyskowych wymaga oprogramowania "serwisowego", obejmującego m.in. procedury reorganizacji zbiorów (mających na celu likwidację nadmiarów – *overflow* – na poszczególnych sektorach), programy "oczyszczania" dysku, dumping na taśmę magnetyczną itp. Koncepcja cylindra (*seek area*) stwarza możliwości optymalizacji (z punktu widzenia czasu dostępu), rozmieszczenia zbiorów dotyczących tego samego programu poprzez lokowanie ich na jednym cylindrze, tj. obszarze poszukiwań. O trudnościach software'owego opracowania pamięci dyskowych świadczy np. słabość dotychczas stosowanych algorytmów randomizacji (nierównomierne obciążenie sektorów) oraz fakt, że dopiero w 10 lat po stworzeniu pierwszej pamięci dyskowej powstała "oficjalna" dyskowa wersja COBOLu, którą z kolei do roku 1967 potrafiła wdrożyć jedynie firma IBM.

A.5 Mówiąc o wzajemnym oddziaływaniu hardware'u i software'u nie sposób pominąć konkretnego rezultatu tych oddziaływań, a mianowicie mikroprogramowania.

Posiadać ono będzie szczególne znaczenie, jako tzw. *firmware*, w maszynach IV generacji, kiedy to użytkownik będzie mógł sam tworzyć wymienne pakiety mikroprogramów (stosując np. specjalne karty dziurkowane o trójkątnych otworach) i w ten sposób dostosowywać niektóre parametry konstrukcyjne do konkretnych zastosowań [A2].

Pierwsze publikacje na temat mikroprogramowania pojawiły się już w 1953 roku [L3]. W 1956 roku na konferencji w MIT przyjęto niejako oficjalną nazwę mikroprogramowanie dla określenia techniki budowy układów logicznych, realizujących operacje

<88>

za pomocą sekwencji impulsów. Każdy impuls można nazwać mikrorozkazem. Mikroprogramy mogą być realizowane "pionowo" tzn. jako sekwencja następujących po sobie mikrorozkazów lub "poziomo", tj. kiedy programista decyduje o otwarciu, odpowiednich bramek poprzez zastosowanie odpowiedniej struktury bitowej w kodach operacji [X16]. Ważną zalecą mikroprogramów jest to, że realizacja ich nie "obciąża" jednostki centralnej, ponieważ działają one niejako automatycznie i co ważniejsze, mogą przebiegać równocześnie (*in one pulse time*).

Mikroprogramy są przedstawiane za pomocą pamięci pasywnych (*read-only memory*, *non-volatile store*, *fixed store*). Jedną z pierwszych maszyn programowanych był EDSAC 11, w którym mikroprogramowanie realizowane było poprzez odpowiednie łączenia w pamięci ferrytowej. Znaczne mikroprogramowanie posiadała maszyna TX-O, zbudowana w Lincoln Laboratories MIT (w maszynie tej wykorzystano dorobek wspomnianej konferencji na temat mikroprogramowania).

Mikroprogramowanie stanowi ważny element ujednolicania architektury maszyn (np. modeli serii IBM 360), wchodząc w skład środków emulacji (rozdział VI, pkt.2).

A.6 Krokiem zmierzającym do wyeliminowania strat pamięci potrzebnej na przechowywanie adresów i ułatwienia techniki programowania. zadań w zakresie wyszukiwania informacji jest propozycja pamięci asocjacyjnej. Koncepcję pamięci asocjacyjnej po raz pierwszy sformułowali prawdopodobnie Newell A, Shaw J.C., Simon H.A. [C3], wykorzystując metodę pośredniego adresowania.

Istnieją różne koncepcje organizacji konstrukcji tego typu pamięci [K1].

<89>

Hardware'owe rozwiązanie pamięci asocjacyjnej posiada znaczne walory w porównaniu z tzw. software'owym typem. pamięci asocjacyjnej. Każda komórka posiada możność wykonania operacji porównania ze z góry zadanymi wartościami (oczywiście bez angażowania urządzenia arytmetycznego). Poszukiwanie określonych wartości odbywa się bez pośrednictwa adresów i wykonywane jest równocześnie we wszystkich komórkach. Rezultat porównań zapamiętywany jest w specjalnych wskaźnikach. Koncepcja działania tego typu pamięci szokuje oryginalnością i efektywnością. Jest s technicznego punktu widzenia trudna do zrealizowania.

Sposób software'owej organizacji pamięci asocjacyjnej polega na zastosowaniu tzw. list adresowych (łańcuchowego adresowania).

Ogólnie rzecz biorąc, pamięć asocjacyjna adresowana jest zawartością, tzn. informacja odszukiwana jest w pamięci na podstawie treści a nie adresów. Poza usprawnianiem procesów wybierania, zalety pamięci asocjacyjnej wykorzystywane być mogą do przyśpieszenia sortowania (dzięki ominięciu pracy sekwencyjnej).

Nie tylko pamięć asocjacyjna jest próbą przełamania adresowości. Podobną rolę spełnia również pamięć stosowa (*pushdown memory*, *stack store*).

B.1. Jedną z podstawowych charakterystyk języków wewnętrznych i symbolicznych jest adresowa struktura rozkazów. Liczba części adresowych rozkazu waha się od jednego do pięciu (najczęściej spotyka się maszyny jedno i dwu-adresowe) i określana jest przez konstrukcję każdej maszyny. Maszyny UNIVAC I, II, seria IBM 700, większość maszyn typu

<90>

"Princeton Class" (a więc opracowanych wg propozycji Johna von Neumanna) były maszynami jednoadresowymi. Inne, takie jak np. UNIVAC Scientific ERA-1103, należały do maszyn dwuadresowych. Nadmienić można, że rozróżnia się dwa rodzaje systemu dwuadresowego: jeden, gdzie ukazywany jest adres argumentu i adres rezultatu, i drugi, tzw. 1+1 adresowy, gdzie ukazywany jest adres argumentu i adres następnego rozkazu. Adresowe rozkazy jeden plus jeden są szczególnie przydatne w maszynach z pamięcią bębnową, ze względu na możliwość usprawnienia wybierania (stosowane były w maszynach IBM 650, ODRA 1003). Konstruowane też były maszyny trójadresowe (NORG, MIDAC, STRIEŁA), czteroadresowe (EDYAC, SEAC), pięcioadresowe (maszyna SAPO-CSR – wybór czwartego lub piątego adresu następował alternatywnie w zależności od tego, czy wynik operacji był dodatni czy ujemny).

Około roku 1960 publikowane były w różnych krajach (w tym również w Polsce) koncepcje tzw. maszyn bezadresowych [P1]. Generalnym założeniem tych maszyn miało być znaczne uproszczenie zarówno konstrukcji jak i programowania. Aczkolwiek poczyniono pierwsze eksperymenty (m.in. w Polsce i Australii) nie doszło do przemysłowej produkcji tych maszyn. Maszyny bezadresowe przeznaczone były w zasadzie do wykonywania prostych i powtarzalnych obliczeń technicznych. Być może idee te zostaną znowu podjęte, jeśli na szerszą skalę zacznie się stosować niezależne moduły asocjacyjnej pamięci operacyjnej.

B.2 Wiadomo, że elektroniczna maszyna cyfrowa to zespół zarówno środków hardware'owych (konstrukcyjnych – w dosłownym tłumaczeniu "żelaznych") i software'owych (programowych – "miękkich"). Dokładna definicja komputera nie została właściwie przez nikogo

<91>

podana, mimo, iż wszyscy mniej więcej zdajemy sobie sprawę, o co chodzi, gdy mówimy o komputerze. Po to, by unaocznić software'owe i konstrukcyjne strony nowoczesnego komputera podajemy jego następującą definicję: "Komputer jest to środek o dużej złożoności konstrukcyjnej (zespół różnych urządzeń) automatycznie wykonujący ciągi operacji arytmetycznych, logicznych, przesyłowych, sterujący pracę własnych urządzeń, sygnalizujący własne błędy, kontaktujący się z operatorem, posiadający własny język porozumiewania się (programowania) i język liczenia dwójkowy, dwójkowo-dziesiętny)".

Jak z powyższych definicji wynika, elementy "zmienne” (software'owe) odgrywają w maszynie decydującą rolę (przynajmniej z punktu widzenia użytkownika). Nie zawsze

jednak tak było.

Pierwsza elektroniczna maszyna cyfrowa ENIAC właściwie komputerem jeszcze nie była. Składała się ona prawie wyłącznie z hardware'u (nawet program nie był przechowywany, m.in. z powodu małej pamięci, składającej się z 20 rejestrów po 10 cyfr). Już przed tą maszyną istniały próby budowy automatycznych przekaźnikowych maszyn liczących, np. MARK I – ASCC (*Automatic Sequence Controlled Calculator*) budowany w Harvard University w latach 1939-1944 pod kierownictwem prof. H.Aikena oraz Complex Computer skonstruowany w Bell Telephone przez dra G.R.Stibitza.

Zainteresowanie software'm znacznie wzrosło dopiero przy maszynach III generacji, wyposażonych w *time-sharing*, wielodostępnych i wieloprogramowych. Jeszcze w 1957 roku udział kosztów software'u w ogólnym koszcie maszyny wynosił [K4] zaledwie 25% zaś na przestrzeni lat 1967 – 1972 ma wzrosnąć do 70%.

<92>

Z technicznego punktu widzenia przyjęto wyróżniać następujące generacje komputerów:

– zerowa – przekaźnikowa (lata czterdzieste),

– pierwsza – lampowa (ENIAC – 1945, ODRA 1001 – 1960, ZAM-2 – 1961)

– druga – tranzystorowa (mniej więcej od 1954 r. – TRADIC).

Rozróżnienie dalszych generacji na podstawie powyższego kryterium jest w zasadzie niemożliwe ze względu na wzrost znaczenia organizacji działania komputerów.

Z programowego punktu widzenia pierwsza generacja charakteryzuje się stosowaniem języków wewnętrznych i symbolicznych prostych (1.1) – zaś w drugiej dochodzą języki wyższego rzędu języki symboliczne z makrorozkazami oraz proste systemy operacyjne.

Nie ukształtowały się jeszcze definicje maszyn III i IV generacji, przytoczymy więc tylko niektóre typowe ich cechy:

III generacja

1. praca w podziale czasu (*time sharing*)

czyli zdolność współpracy – w kolejnych krótkich jednostkach czasu – jednostki centralnej z wieloma urządzeniami wejścia – wyjścia.

2. wieloprogramowość

czyli zdolność realizacji kilku programów znajdujących się równocześnie w maszynie,

3. wielodostępność

czyli zdolność współpracy maszyny z wieloma operatorami, przy czym każdy z nich ma wrażenie, że jest wyłącznym jej użytkownikiem,

4. minimalna pojemność pamięci operacyjnej 32K słów (o ile nie

<93>

ma masowej pamięci pomocniczej o dostępie wyrywkowym); z reguły stosowana jest byte'owa lub znakowa organizacja pamięci (zamiast wyrazowej),

5. praca asynchroniczna

m.in. dzięki budowie modularnej, w której moduły wejścia-wyjścia kontrolują urządzenia transferowe bez angażowania jednostki centralnej,

6. złożony system operacyjny do sterowania pracą asynchroniczną, komunikacji z operatorem itp.

IV generacja

obejmuje cechy III generacji plus:

1. *firmware* stosowany zarówno w emulacji, jaki w programach użytkowych,

2. wielka integracja elektroniczna (LSI – *large scale integration*),

3. rozbudowa pamięci operacyjnej do 1 ml słów (lub 4 ml byte'ów),

4. wzrost stopnia wielodostępności i środków komunikacji człowiek-maszyna (w szczególności opracowanie praktycznych języków konwersacyjnych),

5. wieloprocesorowość oznaczająca zastosowanie kilku jednostek centralnych, (np. cztery jednostki w maszynach IRIS 80 lub K202),

6. wzrost stopnia modularności oznaczający m.in. stosowanie kilku modułów pamięci operacyjnej o niezależnym dostępie, modułów niezależnych (nie powiązanych hierarchicznie) jednostek centralnych, niezależnych modułów wejść-wyjść, itp.

Cechą charakterystyczną IV generacji jest znaczny wzrost szybkości maszyny głównie poprzez ulepszenie organizacji działania

<94>

(a nie zmianę parametrów technicznych poszczególnych układów elektronicznych).

Maszyny IV generacji powinny być wzajemnie bardziej zamienne ze względu na elastyczność organizacji działania: modułowość, wymienne pakiety firmware u, rozwinięte stronicowanie, software do kierowania przepływem danych itp.

Trudno jest w tej chwili mówić o ostatecznym kształcie maszyn IV generacji, tym bardziej, że właściwie nie rozpoczęto jeszcze ich produkcji na skalę przemysłową. Jeszcze parę lat temu generacja ta stanowiła "szyld" tego wszystkiego, co może być ulepszone w komputerach.

W 1967 roku redakcja znanego fachowego czasopisma amerykańskiego *Datamation* [X17] przeprowadziła ankietę na temat maszyn IV generacji. 50% pytanych oświadczyło, że produkcja komputerów tej generacji rozpocznie się w latach 1971-1975, 25% – pod koniec 1970 roku, zaś pozostałe 25% oświadczyło, że komputery te w ogóle ... nie

zostaną wyprodukowane. Prognozowanie rozwoju komputerów jest bardzo zawodne. Oto przykłady:

1. skonstruowanie perceptronu w 1958 roku stało się podstawą przepowiedni o zastąpieniu programowania przez systemy samoorganizujące,

2. kriotrony (1959) miały stać się jedną z podstawowych pamięci,

3. od paru lat mówi się o możliwości wykorzystania światła laserowego w urządzeniach pamięciowych,

4. kiedy w 1953 roku dokonano tłumaczenia z rosyjskiego na angielski, wydawało się, że otwiera się droga do automatyzacji tłumaczeń na szeroką skalę,

<95>

B.3 Jedną z ciekawszych tendencji rozwoju konstrukcji maszyn jest **praca wieloprocesorowa** (w ramach jednej maszyny) i praca **wielomaszynowa**.

Przez system wielomaszynowy rozumiemy zespół kilku (-nastu, -dziesięciu) maszyn pracujących równocześnie i komunikujących się wzajemnie np. za pośrednictwem centralnego systemu operacyjnego, wymieniających dane pomiędzy sobą itp. Granice pomiędzy systemem wieloprocesorowymi wielomaszynowym są czasem trudne do uchwycenia, jeśli rozpatrujemy je jedynie na tle funkcjonalnym i możliwości przetwarzania. Jakimś, w miarę pomocnym kryterium może być rozrzucenie terytorialne poszczególnych jednostek w systemach wielomaszynowych.

W pracy niniejszej używamy pojęcia "system wielomaszynowy" zamiennie do pojęcia "wielosystem" [1] . W publikacji [G5] za system wielomaszynowy uważa się taki rodzaj wielosystemu, w którym wszystkie "maszyny" są równoprawne i posiadają identyczną wydajność (w odróżnieniu od systemów hierarchicznych, w których rozróżnia się maszyny nadrzędne i podrzędne). Na marginesie tej definicji można rzucie uwagę, że maszyny wieloprocesorowe IV generacji najczęściej posiadają właśnie taką organizację działania, a mimo to nie uważa się ich za systemy wielomaszynowe.

[1] *Termin "wielosystem", podany przez autora budzi zastrzeżenia. Zdaniem Redakcji, wystarczy operowanie terminem "systemu wielomaszynowego" (Przyp.Red.).*

Jako przykład systemu wielomaszynowego wymienić można system 29 maszyn pracujących dla IPC (*International Paper Company*), w którym maszyny są rozrzucone w 25 miejscach i połączone siecią transmisji danych.

<96>

Ilustracją maszyny wieloprocesorowej jest ILLIAC IV. Maszyna ta posiada 256 urządzeń arytmetycznych, podłączonych do 4 jednostek sterowania. Dzięki zwielokrotnieniu urządzeń arytmetycznych maszyna ta osiąga szybkość 1 mld oper/sek (1-18).

Część centralna innej maszyny – CDC 6600 – zawiera 10 wyspecjalizowanych jednostek

przetwarzania (m.in. do operacji zmiennoprzecinkowych [D4].

Systemy wielomaszynowe stosowane są wtedy, gdy:

a) wymagana jest praca bezawaryjna, np. przy sterowaniu procesami technologicznymi,

b) wyniki obliczeń muszą być bezbłędne (w tym celu na kilku maszynach-procesorach wykonywane są takie same zadania),

c) występują różnorodne zadania, wykonanie których może być podzielone pomiędzy maszynami,

d) ilość informacji jest tak duża, że gwarantuje pewien stopień obciążenia dla każdej z maszyn. Niekiedy brany jest pod uwagę tylko jeden z powyższych czynników.

Historia systemów wielomaszynowych jest dosyć długa. Już jedna z pierwszych automatycznych maszyn przekaźnikowych BELL V (1944) firmy Bell Telephone Laboratories składała się z dwóch identycznych maszyn, równolegle podłączonych do wejścia, pracujących niezależnie i porównujących wyniki takich samych obliczeń

W zbudowanej w 1958 roku w Czechosłowacji maszynie SAPO zastosowano trzy podłączone równolegle arytmometry, korygujące wzajemnie wyniki (za wynik prawidłowy uważano rezultat uzyskany co najmniej na dwóch arytmometrach).

Projekt SOLOMON, realizowany przez firmę Westinghouse (USA)

<97>

przewiduje zastosowanie około tysiąca niezależnych jednostek liczących równolegle zadany program.

System STRETCH (IBM, 1961) obejmuje dwie maszyny cyfrowe, z których jedna o działaniu szeregowym (tzn. wolniejsza) przeznaczona jest do wykonywania prac przygotowawczych, zaś druga oddziałaniu równoległym wykonuje podstawowe obliczenia.

W latach 50-tych w USA do sterowania środkami obrony przeciwlotniczej stosowano zestawy dwóch maszyn ANFSQ-7 (1956) pracujących równolegle w celu zabezpieczenia wysokiej pewności działania.

Również w USA stosowany jest system Melseng, w skład którego wchodzą 3 maszyny (IBM 1620, IBM 1410, S-C 4020), przeznaczony do prac inżynieryjnych i przedstawiania wyników w postaci graficznej.

Znana jest koncepcja zastosowania kilkunastu emc w kombinacie metalurgicznym Spencer Works do obsługi wielopoziomowego kompleksu produkcyjnego. Maszyny działają na tzw. poziomach ustawionych hierarchicznie: maszyny poziomu I sprawują funkcje nadrzędne w stosunku do maszyn poziomu II itp. Różnicowanie poziomów nastąpiło nie wg etapów przetwarzania danych, lecz poziomów zarządzania. Poziom I przeznaczony był do wykonywania prac związanych z planowaniem wieloletnim i rocznym, poziom II i III zajmował się planowaniem krótkookresowymi harmonogramami

produkcji, zaś poziom IV – sterowaniem procesów technologicznych.

B.4 Jedną z głównych cech nowoczesnego komputera jest praca wielodostępna w podziale czasu. Z punktu widzenia programowania systemy te możemy podzielić na uniwersalne i wyspecjalizowane.

<98>

Systemy uniwersalne są zdolne do realizacji dowolnego programu. Praca maszyny jest programowana bezpośrednio przez użytkownika w terminalu [1] (względnie program jest ściągany z biblioteki), w związku z czym operator w zasadzie nie wie, co maszyna w danej chwili wykonuje.

[1] *Coraz częściej używa się terminu "urządzenie końcowe" (Przyp.Red.).*

Systemy wyspecjalizowane przeznaczone są do wykonywania jednego zespołu programów dla wielu użytkowników. Przykładem takiego systemu może być program rezerwacji miejsc w komunikacji lotniczej. Maszyny pracujące w tych systemach mogą być konstrukcyjnie (mikroprogramowo) przystosowane do określonego rodzaju pracy. Wprowadzenie uniwersalnych systemów było możliwe dzięki osiągnięciom w konstruowaniu i programowaniu maszyn cyfrowych, które z kolei zostały przygotowane przez prace teoretyczne. W 1959 roku na konferencji UNESCO wygłoszony został przez Ch.Strachey'a pierwszy referat na temat idei *time-sharing*-u i pracy wielodostępnej. W dwa lata później J.McCarthy sformułował pięć głównych wymogów podziału czasu:

1. duża ferrytowa pamięć operacyjna,

2. system przerwań,

3. ciągła (*non-stop*) praca maszyny przy realizacji różnych programów, a więc automatyczne przechodzenie od zadania do zadania,

4. ochrona pamięci dla poszczególnych programów,

5. masowa pamięć pomocnicza, przeznaczona do przechowywania zbiorów współpracujących z poszczególnymi programami.

Zanim doszło do pierwszych teoretycznych rozpraw, musiała zostać postawiona sama problematyka. Konieczność zastosowania time-sharingu wyszła w trakcie budowy systemu obronnego SAGE

<99>

(*Semi-Automatic Ground Environment*) prowadzonej przez Lincoln Laboratory MIT i Rand Corp.

W listopadzie 1961 roku zademonstrowano w ośrodku obliczeniowym MIT jeden z pierwszych uniwersalnych systemów, który nie spełniał jednak wszystkich pięciu wymogów Mc Carthy'ego. W 1963 roku znane były dwa zaawansowane wielodostępne systemy uniwersalne. Jeden z nich narodził się w MIT, gdzie opracowano projekt MAC (Multipie-Access Computer) i uruchomiono system CTSS (*Compatible Time-Sharing System*) na maszynie IBM 7090. W tym samym roku opracowano system w System Development Corp.

W roku 1964 działało w USA około dziesięciu systemów wielo dostępnych. W roku 1965

pojawiają się pierwsze seryjne maszyny spełniające wszystkie wymogi – GE 265 i UNIVAC 491.

Specjalizowane systemy wielodostępne niekoniecznie muszą pracować w podziale czasu. W 1952 roku w American Airlines uruchomiono system rezerwacji miejsc, w którym użyto specjalizowanej maszyny z pamięcią bębnową z wbudowanymi programami. Wśród systemów specjalizowanych rozróżnia się tzw. rozwinięte systemy, które, mimo ograniczeń związanych z danym rodzajem pracy, mogą być modyfikowane w zależności od potrzeb każdego użytkownika (np. systemy informacyjno-diagnostyczne w szpitalnictwie, systemy wybierania informacji itp.).

Rozwój zastosowań systemów wielodostępnych zależny był m.in. od wprowadzenia odpowiednich języków "konwersacyjnych", innych od używanych w partiowo-okresowym przetwarzaniu.

Pionierskim osiągnięciem był tutaj JOSS, opracowany w RAND CORP. Mniej więcej w tym samym czasie (1964) powstał język BASIC autorstwa J.Kemeny i T.E. Kurtz z Dartmouth College.

<100>

Do tej pory języków konwersacyjnych opracowano wiele. Pewne informacje na ten temat można znaleźć w rozdziale IV.

VIII. Rozwój programowania a zastosowania elektronicznych maszyn cyfrowych

1. Uwagi ogólne o przeszłości i perspektywach zastosowań.

Jeden z ojców przemysłu komputerowego John Mauchly twierdził, że tylko 4-5 wielkich koncernów USA potrafi zagospodarować komputery. Tymczasem już w 1951 roku pracowało około 100 komputerów, zaś obecnie jest ich tysiąc razy więcej. Jest to skok ogromny, zważywszy, że komputer to nie liczydło biurowe za kilkadziesiąt złotych, lecz drogi zestaw urządzeń przeciętnie za kilkaset tysięcy dolarów. Mamy więc do czynienia z urządzeniem, które co prawda bezpośrednio w zasadzie nie tworzy dóbr produkcyjnych (materialnych) ani (przynajmniej dotychczas) nie stanowi przedmiotu prywatnego użytku szerokich rzesz obywateli, ale stało się przedmiotem działania najnowocześniejszych gałęzi przemysłu i miernikiem poziomu technicznego krajów.

Upraszczając zagadnienie, można powiedzieć, że już po pierwszej wojnie światowej Istniały pewne przesłanki do zbudowania i zastosowania komputera. Znany był zapis magnetyczny (stanowiący podstawę urządzeń pamięciowych), wynaleziono lampy elektronowe oraz tak ważny układ, jakim jest przerzutnik. Babbage podał funkcje automatycznej maszyny liczącej, zaś Boole sformułował aparat formalny do opisu jej

działania. Istniało duże zapotrzebowanie na obliczenia w astronomii i ilościowej analizie chemicznej.

<101>

Mimo to, dopiero pod wpływem potrzeb II wojny światowej rozpoczęto prace nad budową uniwersalnej szybkiej maszyny liczącej. Co prawda, ENIAC nie zdążył już spełnić swojego przeznaczenia (obliczanie torów balistycznych), ale okazało się, że z powodzeniem może rozwiązywać inne problemy matematyczne.

Pierwsze komputery służyły głównie do obliczeń naukowych i technicznych. Problematyka administracyjna, w której występuje masa informacji i różnorodne reguły postępowania, nie mogła być na tych maszynach przetwarzania z powodu braku dużych pamięci i szybkiego aparatu piszącego.

Od roku 1949 równolegle w USA i Wielkiej Brytanii prowadzone były prace nad zbudowaniem odpowiednich maszyn. W dwa lata później powstały komputery LEO (W.B.) i UNIVAC (USA), które zapoczątkowały epokę naprawdę uniwersalnych maszyn matematycznych. Komputery te (podobnie jaki późniejsze) mogły rozwiązywać zarówno skomplikowane zadania matematyczne, jaki złożone problemy administracyjne (w rodzaju gospodarki materiałowej, płac itp.)

Maszyna LEO zbudowana została przez dużą firmę gastronomiczną (!) dla własnych potrzeb, a mianowicie do obsługi kilkuset restauracji, piekarń i kawiarń w zakresie ewidencji sprzedaży, badania popytu, obliczania płac itp. Maszynę oparto o rozwiązania konstrukcyjne maszyny EDSAC a koszt jej wynosił około 75 tysięcy funtów. Nie była to więc kwota wysoka.

Maszyna UNIVAC po raz pierwszy wykorzystywana była przez Biuro Spisu Ludności USA. Obliczono, że w pracach tego biura jedna minuta pracy komputera równoważna była pod względem wydajności 67 godzinom ręcznej pracy. UNIVAC I był pierwszym komputerem wprowadzonym do produkcji przemysłowej i używanym do wielu różnorodnych prac (do sporządzania list płac, ewidencji

<102>

materiałów, rozliczania ubezpieczeń, ewidencji i analizy sprzedaży itp.

Lista zastosowań komputerów jest bardzo szeroka. Obejmuje obecnie 1700 dziedzin techniki, ekonomiki, nauki i kultury. 80-90% komputerów znajduje zastosowanie w przedsiębiorstwach, instytucjach bankowych i handlowych, centralnych urzędach rządowych itp.

Obserwujemy w ostatnich latach tendencję odstępowania od jednotematycznych rozwiązań fragmentarycznych na rzecz wielotematycznych systemów zintegrowanych, w których tworzony jest wspólny bank danych. Systemy te mają na celu bieżące informowanie kierownictwa, zapewniając równocześnie najbardziej efektywne przetwarzanie (poprzez specjalne procedury operowania na wspólnej bazie danych).

Stosunkowo rzadko używane są komputery do sterowania procesami technologicznymi. Mimo, iż są to zastosowania z reguły bardzo opłacalne (w masowym typie produkcji o

wysoce zautomatyzowanych procesach), wymagają długofalowych prac "identyfikacyjnych" i są bardzo odpowiedzialne. Musi być dokładnie sformułowany proces technologiczny (łącznie ze wszystkimi możliwymi odchyleniami), należy zastosować dokładną aparaturę kontrolno-pomiarowa i wysokiej jakości komputery (długi okres pracy międzyawaryjnej).

Pierwsze zastosowanie komputera do sterowania oddziałem produkcyjnym w tzw. układzie zamkniętym miało miejsce w 1959 roku w rafinerii Port Arthur (USA).

W krajach zachodnich w warunkach ostrej walki konkurencyjnej komputery są często używane do obsługi klientów, szczególnie w sprzedaży hurtowej. Punkty sprzedaży połączone są siecią transmisji danych z ośrodkiem obliczeniowym. Komputer wydaje polecenie realizacji zamówienia na punkt z lokalizowany najbliżej siedziby klienta i posiadający wymagany asortyment.

<103>

Od kilkunastu lat stosuje się komputery do załatwiania rezerwacji miejsc w komunikacji lotniczej. Problem polega na tym, że pasażerowie dokonują wstępnej rezerwacji miejsc i potem często ją zmieniają w ostatniej chwili. Towarzystwa lotnicze mogłyby z tego powodu ponosić straty (niewykorzystane miejsca trudno jest klasycznymi środkami komunikacyjnymi rozprowadzić szybko pomiędzy wiele punktów sprzedaży). Kasy biletowe podają zgłoszenia (lub odwołania) rezerwacji poprzez sieć transmisji do komputera. Otrzymują odpowiedź już po kilku sekundach.

O uniwersalności komputerów świadczy fakt używania ich do planowania produkcji kwiatów w dużych kwiaciarniach. Kwiaciarnie takie dostarczają kwiaty do wielu krajów, ale nie chodzi tutaj o obsługę rozliczeń finansowych. Problem polega na tym, że kwiatów nie można magazynować, np. chryzantemy muszą być rozesłane do klientów samolotami najpóźniej po 3 tygodniach od momentu ich posadzenia. Komputery kontrolują więc wielkość zapasów i wykonują obliczenia w zakresie prognozowania ich zbytu.

Maszyny matematyczne używane są również do tłumaczenia tekstów z języka na język. Już w styczniu 1954 roku publicznie demonstrowano w Nowym Jorku tłumaczenie z rosyjskiego na angielski. W ZSRR opracowano m.in. programy tłumaczenia z francuskiego i angielskiego na rosyjski. W Japonii zbudowano komputer, który nie tylko tłumaczy z angielskiego na japoński, ale jednocześnie po japońsku czyta przetłumaczone zdania.

<104>

Komputery stosuje się dosyć szeroko w lecznictwie. 250 ekspertów z 22 krajów na naradzie zwołanej z inicjatywy międzynarodowej organizacji przetwarzania informacji IFIP orzekło optymistycznie, że już w 1980 roku większość lekarzy będzie miało dostęp do komputerów w celach konsultacji i diagnostyki.

Ostatnio komputery coraz częściej występują w roli "twórców" muzyki, poezji i malarstwa. Z uwagi na awangardowość sztuk nowoczesnych zdarzają się przypadki nieodróżniania utworu maszynowego od dzieła człowieka.

Uznanie zdobywają komputery w systemach nauczania. Maszyna przerasta nauczyciela szybkością reakcji, zdolnością zapamiętania i wybierania ogromnych ilości informacji, oraz – co najważniejsze – nie męczy się. To, czy komputer zastąpi uczniowi wartości wychowawcze i potrzeby psychiczne, wynikające z bezpośredniego kontaktu ucznia z nauczycielem, to już inna sprawa.

Szczególnie dobrze spisują się komputery w szkoleniu programistów. Ponieważ stale odczuwa się deficyt tego rodzaju kadr, w krajach zachodnich szkoli się w tym zawodzie nawet niewidomych i więźniów. Tak na przykład, więzienie waszyngtońskie posiada własną szkołę programowania". Wyselekcjonowanym więźniom stworzono swobodniejsze warunki (cele i sale wykładowe poza głównym budynkiem więziennym, odpowiednia atmosfera w czasie wykładów). Mieli oni zapewniony stały kontakt z komputerem IBM 360/40 za pośrednictwem terminali. W okresie rocznej nauki, oprócz kilku języków programowania, więźniowie opanowali również podstawy księgowości, ekonomiki i przetwarzania informacji. Absolwenci w okresie odbywania kary otrzymywali zlecenia na prace programowe od instytucji rządowych, zaś po zwolnieniu

<105>

z więzienia mogli podjąć studia specjalistyczne lub otrzymywali pracę programisty w rządowych ośrodkach obliczeniowych.

Eksperci przewidują, że ekspansja komputerów trwać będzie nadal. Być może powstaną światowe sieci komputerów, obsługując międzynarodowe instytucje gospodarcze, handlowe, prawnicze, lekarskie i policyjne. Coraz bardziej sam człowiek będzie poddawany działaniu komputerów. Eksperci firmy Rand Corp. twierdzą, że po roku 1990 nastąpi bezpośrednie podłączenie maszyny matematycznej do mózgu człowieka, zaś jeszcze do roku 2000 powstanie możliwość uczenia się poprzez bezpośrednie utrwalanie informacji w komórkach mózgowych (czyli do końca życia będziemy mogli pamiętać urazy !).

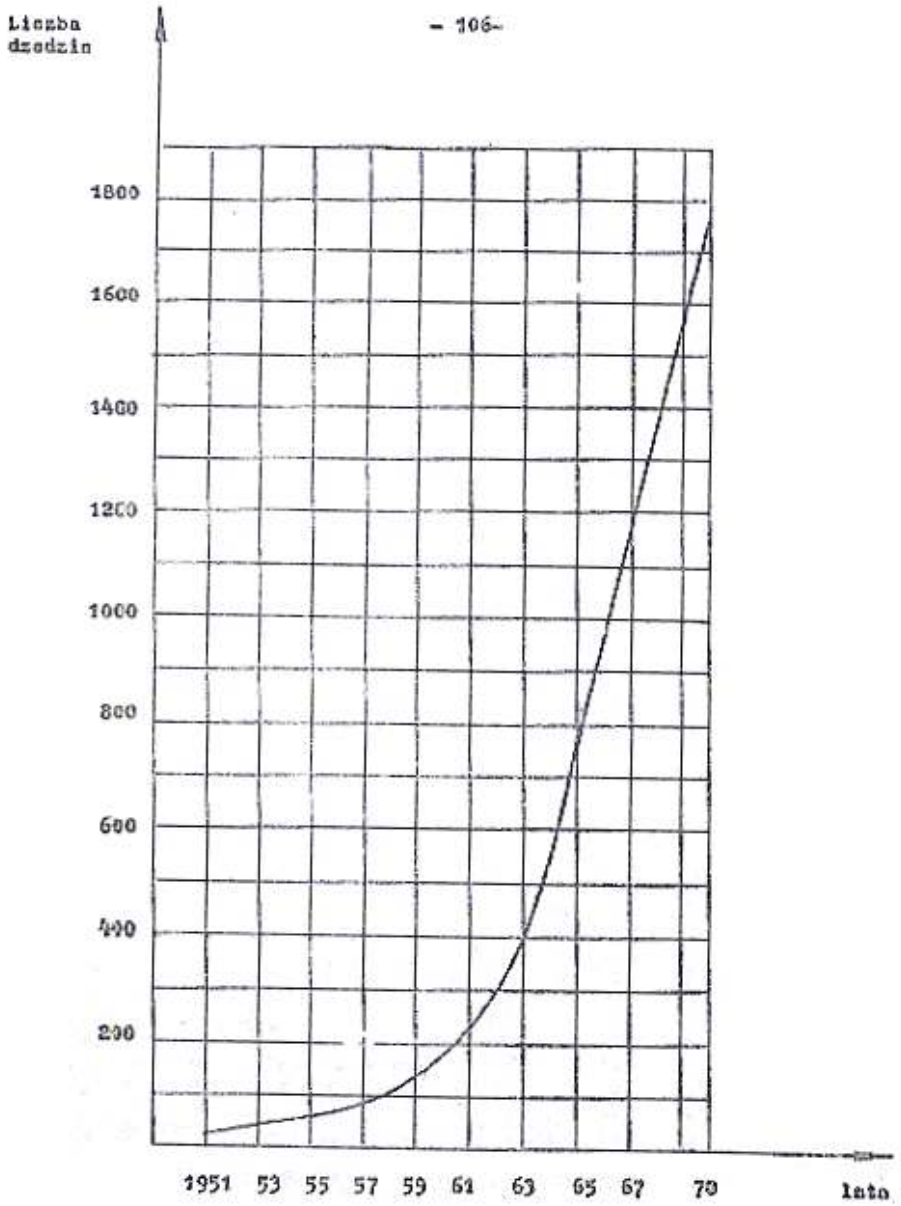
Tyle dywagacji na pograniczu futurologii i fantazji.

2. Statystyka i klasyfikacja zastosowań

Postępy w programowaniu były ściśle związane z problematyką zastosowań. Dopiero po nagromadzeniu kilkuletnich doświadczeń w zastosowaniach problemowych powstały języki problemowe wyższego rzędu: COBOL, ALGOL itp.

W naszych rozważaniach pomocny będzie poniższy wykres: Dynamika wzrostu liczby dziedzin zastosowań komputerów".

<106>



RYS.1.
Dynamika wzrostu liczby zastosowań
/Z - 5 i inne źródła /

<107>

Jak widać s wykresu, zdecydowany wzrost liczby dziedzin przypada na lata 60-te. Można to tłumaczyć tym, że pod koniec lat pięćdziesiątych opracowane zostały podstawowe języki programowania. FORTRAN, ALGOL, COBOL, dzięki czemu przedstawiciele różnych dziedzin mieli łatwiejszy dostęp do komputerów. Z kolei użytkownicy, odsuwając specyfikę własnych zastosowań na tle powyższych, dalekich od doskonałości języków, proponowali własne języki wąskospecjalizowane. Tak więc różnorodność zastosowań znajdowała odbicie w różnorodności języków programowania (1963 – 300 języków, 1966 – 1200 języków). Wprowadzanie języków uniwersalnych (w rodzaju PL/I) powinno zahamować imponujący wzrost liczby języków programowania.

Wg oceny Diebolda (podanej w [K4]) z 1969 roku ewolucję systemów informacyjnych można przedstawić następująco:

Rok 1964 –	druga generacja zastosowań Zakres: administracja i rachunkowość. Kryterium oceny (cel): redukcja etatów, zmniejszenie kosztu.
Rok 1968 –	trzecia generacja zastosowań Zakres: informacja nadzoru (*supervisory information*) Kryterium: zmniejszenie zapasów, stabilizacja obsady personalnej, usprawnienie obsługi klientów, kontrola kosztów.
Rok 1975 –	trzecia i czwarta generacja zastosowań Zakres: informacje dla kierownictwa średniego szczebla (*middle management*) oraz planowanie na szczeblu taktycznym, Kryterium: preliminowanie marketingu, skrócenie okresu zwrotu nakładów optymalizacja obciążenia urządzeń produkcyjnych, bardziej realistyczne prognozy.
Rok 1985 –	piąta generacja zastosowań Zakres: obsługa naczelnego kierownictwa i planowanie na szczeblu strategicznym. Kryterium: planowanie produkcji, zapotrzebowanie na kapitał, planowanie zapasów, siły roboczej itp.

<108>

Typowym poziomem zastosowań komputerów w latach 50-tych i 60-tych były systemy tematyczne, projektowane pod kątem potrzeb działania operatywnego kierownictwa poszczególnych komórek organizacyjnych przedsiębiorstwa (patrz rys. 2).

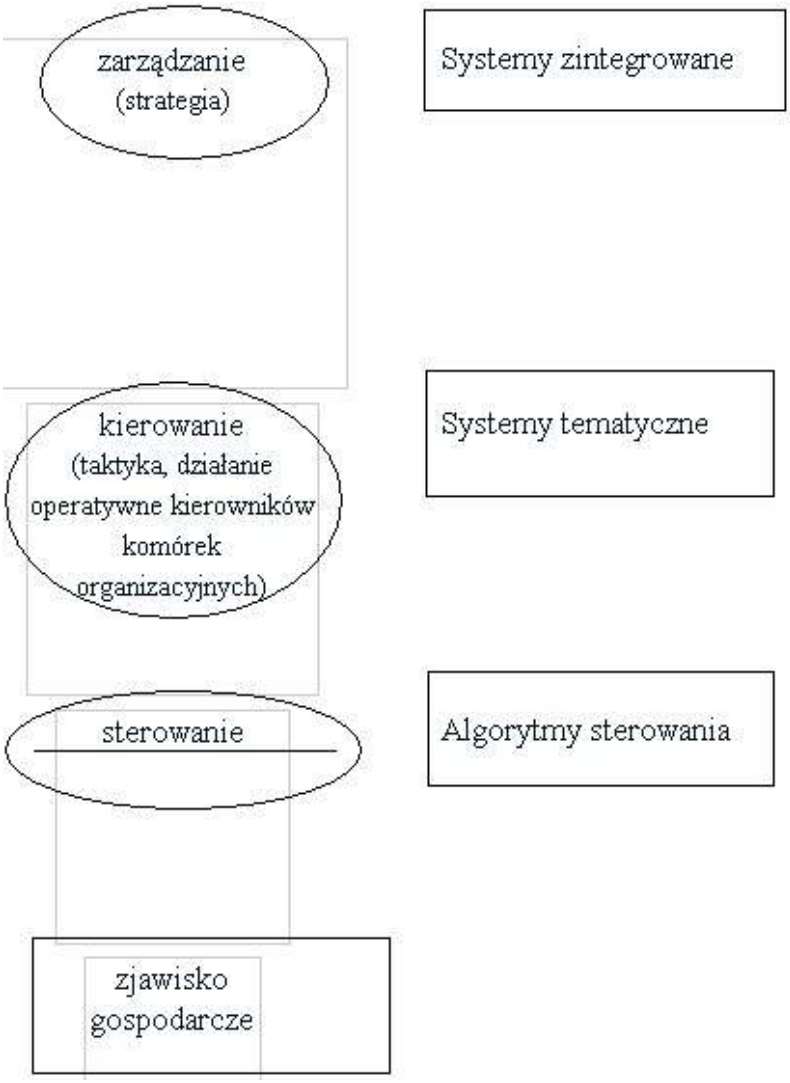
Systemy te można było stosunkowo łatwo opracować i wdrożyć, ponieważ opierano jego doraźne potrzeby ewidencji i sprawozdawczości. Wobec trudności modelowania zarządzania, braku technologiczno-technicznej bazy systemów zintegrowanych (w szczególność i software'u do operowania na wspólnej bazie danych) oraz trudności organizacyjnych, problematyka systemów zintegrowanych stanowiła raczej przedmiot teoretycznych rozważań niż rozwiązań praktycznych. Podstawową cechą systemów zintegrowanych jest integracja Informacji (w postaci tzw. wspólnej bazy danych). Pełny system zintegrowany charakteryzuje się tym, że integracji informacji towarzyszą:

a) integracja organizacji (złamanie sztywnych struktur wydziałowych, utrudniających wykorzystanie wspólnej bazy danych i integrację funkcjonalną),

b) integracja środków technicznych (zastosowanie oprócz ETO różnorodnych środków małej i średniej mechanizacji,

<109>

Rys. 2. Szczeble działalności i odpowiadające im poziomy przetwarzania informacji.



<110>

urządzeń transmisji danych t dostatecznie rozbudowana sieć telefoniczna i dalekopisowa),

c) integracja użytkownika informacji ze środkami technicznymi (obsługa terminali, opanowanie języków konwersacyjnych),

d) integracja funkcjonalna (funkcje poszczególnych komórek podporządkowane wspólnemu celowi).

System zintegrowany w skali przedsiębiorstwa nazwać można podstawowym. Do zintegrowanych systemów poziomu wyższego proponuje się zaliczyć:

– obsługę kompleksu produkcyjnego obejmującego kooperantów,

– integrację sfery zarządzania (zastosowań ekonomiczno-administracyjnych) ze sferą sterowania procesami technologicznymi, zapewniającą bezdokumentacyjne wykorzystywanie w EPD danych odbieranych przez aparaturę kontrolno-pomiarowa,

– obsługę koncernu (zjednoczenia),

– systemy wieloszczeblowe handlu (obejmujące centrale zbytu, hurt,

detal),

– systemy międzybranżowe (np. obejmujące przemysł konsumpcyjny i handel),

– systemy, w których badania operacyjne są zintegrowane z klasycznym przetwarzaniem danych (w zakresie danych, procedur i funkcji).

<111>

IX. Języki programowania opracowane w Polsce

Do niedawna właściwie ani nie produkowaliśmy ani nie eksploatowaliśmy maszyn bogato oprogramowanych. W krajowych publikacjach (poza pewnymi skryptami i ogólnikowymi artykułami) nie popularyzowano naszych osiągnięć w dziedzinie programowania. Tymczasem "coś niecoś" uzbierało się przez okres minionego dziesięciolecia. Nie pretendując do kompletności, w oparciu o fragmentaryczne dane, postaramy się przekazać pewne wiadomości o rodzimym dorobku w zakresie języków programowania.

1.

Maszyną krajowej produkcji, posiadającą względnie oryginalne oprogramowanie, jest ZAM 41, wyposażony we własne języki: SAS (System Adresów Symbolicznych), PJES (Podstawowy Język Symboliczny), EOL (Expression Oriented Language) oraz wersje językowe ALGOLu, COBOLu i podzbioru GPSS (CEMMA 2 – Cyfrowe Modelowanie Maszyny Analogowej). Oprogramowanie to opracował w II połowie lat sześćdziesiątych (z wyjątkiem SAS, którego podstawy powstały wcześniej) Instytut Maszyn Matematycznych w Warszawie. Pierwszym językiem wyższego rzędu opracowanym w tym Instytucie był język SAKO (System Automatycznego Kodowania), którego pierwszy translator uruchomiono w 1962 roku. SAKO był w znacznej mierze językiem oryginalnym oraz posiadał polską terminologię, np. zwroty:

GDY BYŁ NADMIAR: α INACZEJ β
SKOCZ WEDŁUG I: α β ...
WRÓĆ itp.

Stosowano w nim zapis algebraiczny, np. wyrażenie
X = (-B – √D) / 2A

było zapisywane jako
X = (-B – PWK (DELTA)) 2 x A

Język EOL (Expression Oriented Language) jest językiem przeznaczonym

<112>

do przetwarzania symboli, w szczególności nadającym się do pisania translatorów (np. użyto go do pisania translatora COBOLu dla ZAM 41). Pierwsze wersje języka EOL powstały pod kierownictwem prof. dr Leona Łukaszewicza w Instytucie Maszyn Matematycznych w latach 1965-1966. Wersję EOL-3 opracowano w roku 1967 na uniwersytecie Illinois (USA) i zrealizowano w roku 1968 na maszynach IBM 7094 i IBM

360 [L5], [L6]. W języku EOL wykorzystano niektóre idee występujące w innych językach do przetwarzania symboli (COMIT, IFL-V). Program może być pisany w wersji polskiej i angielskiej, z tym, że w ramach jednej sekcji należy stosować wyłącznie słowa kluczowe jednego języka.

Oto przykłady niektórych instrukcji języka EOL.

WSTAW (*put*) oznacza pobranie wyrażenia *E<n>* z pamięci roboczej i umieszczenie go w *P<m>* w pamięci plikowej.

POBIERZ (*get*) oznacza pobranie z *P<n>* i zapisanie do *E<m>*

ZAMIEŃ (*exchange*) oznacza zamianę pomiędzy sobą wartości wyrażeń lub plików, wskazanych przez dwa argumenty rozkazu.

Wersja COBOLu dla ZAM 41 obok szeregu udogodnień zawiera cechy oryginalne, które stanowią o jej niezamienności w stosunku do wersji standardowych. Na dobro wersji zaliczyć trzeba uwzględnienie niepozycyjnego zapisu na taśmie dziurkowanej (z użyciem ograniczników pozycji), wygodne komponowanie tekstów są pomocą znaków specjalnych *.SP .LT .YK OHP*, różnorodne sposoby rozpoznawania typu rekordu (wg ilości znaków, ilości pozycji z wykorzystaniem TALLY), zastosowanie tak nowoczesnych instrukcji jak EXAMINE, itp. Elementami decydującymi o niezamienności są

<113>

deklaracje *FIXED*, *FLOAT* inspirowane przez konstrukcję maszyny (wyrazowa organizacja pamięci) i zapewne przez PL/I. Deklaracje te eliminują użycie klauzuli *PICTURE* (można je używać jedynie, gdy nie występują w opisie danej zwroty *FIXED* lub *FLOAT*.

FIXED (*FIXED 2*) oznacza liczby całkowite (długie) w postaci binarnej, zaś *FLOAT* – liczby zmiennoprzecinkowe też w postaci binarnej.

W połowie lat 60-tych powstała w Instytucie Maszyn Matematycznych koncepcja języka LOGOL (dla maszyny ZAM-2), przeznaczonego do analizy językowej, a w szczególności do tłumaczeń z dowolnego języka o znanej gramatyce na dowolny inny język.

2.

Opracowano w naszym kraju szereg translatorów języka ALGOL [L1]:

a) ALGUM dla UMC1 i UMC10, 1965, Centralny Ośrodek Badań i Rozwoju Techniki Kolejnictwa,

b) ALGOL 60 dla ZAM 41 – 1968, dla ZAM 21 – 1966, IMM,

c) MIN AL – podzbiór ALGOLu 60 dla emc ODRA 1204, 1968, Centralny Ośrodek Badań i Rozwoju Techniki Kolejnictwa.

d) ALGOL 1204 – dla emc ODRA 1204, 1970, Katedra Metod Numerycznych Uniwersytetu Wrocławskiego.

3.

Dla maszyn ODRA 1003i 1013 opracowano język MOST, oparty o języki ALGOL i MARK I. Język ten, autorstwa grupy matematyków Katedry Metod Numerycznych Uniwersytetu Wrocławskiego i Wrocławskich Zakładów Elektronicznych ELWRO, dostosowany został do specyfiki konstrukcyjnej ODRY 1003. Wyraża się to w możności użycia klawiatury akumulatora do drukowania śladu programu, adresowaniu z przeplotem (co siódme komórki na bębnie mają przydzielone sąsiednie adresy) poprzez wciśnięcie przycisku ZR na pulpicie

<114>

sterowania. Diagnostyka kompilatora jest dosyć uboga, zaś sygnalizacja błędów następuje poprzez wydrukowanie na dalekopisie "znaku" oraz tekstu instrukcji, w której został wykryty błąd. Znane są dwie wersje języka MOST: MOST 1 (dla emc ODRA 1003) oraz MOST F (dla emc ODRA 1013 uwzględniający zalety pamięci ferrytowej). Jako ciekawostkę podajemy, że dla maszyny ODRA 1013 opracowano translator języka ALGOL (Czechosłowacja, Pilzno) oraz translator języka FALA 68 (Zakład Metod Numerycznych Uniwersytetu MCS w Lublinie). Język FALA 68 stanowi rozwinięcie kompilatora ALGUM i charakteryzowany jest jako język o stopniu kompilacji pośrednim pomiędzy MOSTem i ALGOLem. Zwroty tego języka mogą być pisane w języku polskim lub angielskim (z tym zastrzeżeniem, że nie wolno ich mieszać w tym samym programie). Translator bierze pod uwagę 5 pierwszych znaków nazw danych. Deklaracja *REAL* oznacza liczby zmiennoprzecinkowe zarówno w postaci normalnej (ciąg cyfr przedzielonych kropką dziesiętną) oraz w postaci półlogarytmicznej (z wykazaniem cechy i mantysy).

<115>

X. Ocena stanu dotychczasowego i tendencje rozwojowe programowania

1. Wydaje się, że lata obecne zamykają pierwszy etap rozwoju programowania. Cechą widoczną tego etapu jest nadprodukcja języków programowania przy równoczesnym braku teoretycznych podstaw budowy języków (ujmujących zarówno składnię, jak i semantykę oraz pragmatykę). Dotychczas stosunkowo wiele zbadano jedynie w zakresie składni. Efektem niedorozwoju teoretycznego jest niski stopień zamienności języków, osy też nawet ich poszczególnych wersji, oraz trudności związane z symulacją i emulacją.

Dorobkiem dotychczasowych prac są przede wszystkim języki powszechnego użycia: FORTRAN, ALGOL i COBOL. Przyszłość przyniesie nam zapewne poważne prace teoretyczne, które umożliwią skonstruowanie efektywnych języków, prawdopodobnie wyspecjalizowanych (na problem, nie zaś na maszynę) lecz opartych na wspólnych podstawach teoretycznych. Dotychczas obserwowaliśmy przeważnie przypadkowe (z wyjątkiem ALGOLu i COBOLu) wysiłki poszczególnych producentów czy użytkowników ukierunkowane na tworzenie nowych języków. Większość tych języków miała krótki żywot, a często jedną (czy też jedyną) ich zasługą było wprowadzenie nowej terminologii dla już uprzednio zdefiniowanych spraw. W sytuacji, kiedy mamy kilka tysięcy języków, a prawie każdy z nich operuje innymi pojęciami, bardzo jest trudno znaleźć fachowca, który potrafiłby zapamiętać wszystkie różnice i spróbował wprowadzić wspólny aparat pojęciowy, umożliwiając tym samym szeroką analizę porównawczą.

Przykłady nadmiaru pojęć mogą być następujące:

<116>

– do określenia segmentu programu: *segment*, *program-unit*, *module*

– do określenia deklaracji: *declarative*, *klauzula*, *dyrektywa*.

2. Wydaje się, że w perspektywie można mówić o dwóch tendencjach rozwojowych programowania. Po pierwsze, będą opracowane metajęzyki, czyli języki teoretyczne, a celem ich będzie stworzenie kryteriów oceny języków oraz reguł budowy syntaktyki i semantyki. Po drugie, na bazie języków teoretycznych powstaną język i wąskospecjalizowane, dostatecznie efektywne a równocześnie łatwe w użyciu. Języki te powinny być zbliżone do języków naturalnych (unikając równocześnie werbalności), by umożliwić łatwy kontakt z maszyną, oraz być na tyle "fachowo", by zapewnić łatwe wyrażenie specyfiki problemu. Dostateczna efektywność programu powinna być zapewniona przez środki software'owe budowane modularnie w celu możności stosowania dowolnej kompozycji modułów i dalszej rozbudowy.

Dotychczasowe koncepcje języków [X20] oparte były o tzw. budowę pozycyjną danych, wskazywaną przez specjalne deklaracje i "poziomowanie" hierarchii danych. Przewiduje się, że w przyszłości budowa pozycyjna zostanie zarzucona na rzecz oceny semantycznej.

Za próbę stworzenia metajęzyka można zapewne uważać uniwersalny generator HELP (*Highly Extendible Languages Processor*) ogłoszony przez organizację *Advanced Computer Techniques* i nazwany *general-purpose natural language generator* [X19]. HELP posiada następujące możliwości:

a) translacja każdego języka naturalnego na inny,

<117>

b) translacja skrótowych wersji FORTRANu, COBOLu (np. COAX) i innych języków na formy żądano do kompilacji,

Teoretycznie programista może stosować więc dowolny (nawet własny) język programowania.

3. Niekiedy spotkać można w publikacjach (np. [B11]). Wzmianki o tzw. palindromicznym (odwrotnym) programowaniu. Polega ono na tym, że program wykonywany jest nie od pierwszej lecz od ostatniej instrukcji, dając ten sam rezultat (podobnie jak czytanie słów "radar", "rotor").

Oto inne przykłady pracy palindromicznej.

a) translacja programu wewnętrznego na program w języku zewnętrznym (symbolicznym, wyższego rzędu),

b) procedury wykrywania błędów idąc "od tyłu" (tj. od wyniku),

c) pakiety programowe FLOWCHART przeznaczone do tworzenia schematów blokowych napisanych programów.

4. Jedną z cech charakterystycznych maszyn IV generacji ma być dobre oprogramowanie komunikacyjne. Należy przypuszczać, że w związku z tym znacznie

wzrośnie rola języków konwersacyjnych oraz że klasyczne języki programowania zostaną odpowiednio zmodyfikowane (wzbogacone). Jest to warunek użytkowania wspólnych baz danych w systemach bieżącego informowania kierownictwa.

Tak np. ostatnio trwają prace nad włączeniem do COBOLu właściwości CCF (*COBOL Communication Facility*), która ma umożliwić stosowanie COBOLu w systemach pytanie-odpowiedź. Między innymi będzie to możliwe dzięki oprowadzeniu do DATA DIVISION specjalnej sekcji COMMUNICATION SECTION, instrukcji *RECEIVE*, *SEND*, itp.

W systemach komunikacyjnych dużą rolę powinny odgrywać

<118>

tzw. przyspieszone kompilatory (*incremental compilers*)

[1] *Jak już wspomniano, tzw.przyspieszone kompilatory łaczą w sobie technikę kompilacji i interpretacji. (Przyp.2010)*

[R8], które tłumaczą każde zdanie programu zewnętrznego natychmiast po wprowadzeniu z terminalu *on-line*, dając od razu diagnostykę błędów. W trakcie wprowadzania programu można eliminować poprzednie zdania, wprowadzać dodatkowe zdania itp. Do tego typu translatorów zalicza się: QUIKTRAN, RUSH (wersja PL/I), FIV (wersja FORTRANu IV), CAL oraz w dużej mierze BASIC.

5. Pewną nadzieję (już od pewnego czasu zresztą) na znaczne uproszczenie programowania stwarzają języki tablicowe (*tabular languages*), przeznaczone do tworzenia programów na podstawie odpowiednio opisanych tablic decyzyjnych. Skonstruowano już parę takich języków (TABSOL, LOGTAB, FORTAB, DETAB 165, GECOM), żaden jednak nie uzyskał szerszego zastosowania (podobnie, jak i same tablice decyzyjne).

Wygodne jest stosowanie tzw. przedtranslatorów, pozwalających na budowę programów mieszanych, złożonych z tablic decyzyjnych i z rozkazów COBOLu (lub FORTRANu).

Tablice decyzyjne zostały po raz pierwszy użyte prawdopodobnie w latach 50-tych firmie GE [X21]. W latach 60-tych rozwinięto szersze prace nad tablicami decyzyjnymi (np. *Decision Tables Symposium* w Nowym Jorku w 1962 roku).

Tablice decyzyjne są szczególnie pomocne do przedstawiania złożonych logicznie sytuacji, które jest trudno wyrazić graficznie na schematach blokowych. Inną zaletą jest to, że umożliwiają skontrolowanie, czy uwzględniono wszystkie możliwe kombinacje warunków i wynikające z nich czynności.

<119>

Docelową perspektywą programowania są systemy samoprogramujące. Podobno ma być to cecha charakterystyczna maszyn V generacji.

Wydaje się nam, że do wyeliminowania programistów nie dojdzie nigdy.

<120>

Bibliografia

[A1] d'Agapeyeff A. An Introduction to Commercial Compilers, Computer Analysis and Programmers Ltd, London, 1964 pp.199-253

[A2] Auerbach I.L. Technology and the Future, *Data Systems* 1/71 s.12-13

[A3] Amdahl G.M., Amdahl L.D. Fourth-Generation Hardware, *Datamation* 1/67 s.25-26

[B1] Bowdon B.V. Faster than Thought, 1953

[B2] Booth A.D., Booth K.H.V. Automatic Digital Calculators, London, 1956

[B3] Bielecki J. Maszyna Turinga, *Maszyny Matematyczne* 1/67 s.28-29

[B4] Buxton J. N. (redakcja) Simulation Programming Languages, North-Holland, 1968

[B5] Blau H. In welcher Sprache soll programmiert werden?, BTA 1/70, s.8-9

[B6] Brandon D.H. Management Standards for Data Processing, 1963

[B7] Bernard S.M. The case COBOL, *Computers and Automation* 2/67, s.40-42

[B8] Borowiec J. Język PL/I. Nowe cechy i elementy wyższych języków programowania, *Biuletyn IMM. Nowości Techniczne*, 1/67, s.17-33

[B9] Borowiec J. Wprowadzenie do języka PL/I [w:] Problemy przetwarzania informacji, t.1, WNT, 1970

[B10] Bromberg H. The COBOL Conclusion, *Datamation* 13/67, s.45-50

[B11] Bernstein W.A. Palindromie Programming, *Datamation* 12/69, s.123-124

[C1] Cutler D.I. Introduction to computer programming, Prentice Hall, 1964

[C2] McCracken D.D. Digital Computer Programming, John Wiley and Sons Inc. 1957; wyd. polskie: Programowanie maszyn cyfrowych, PWN, 1962

[C3] Carr J.W. Lectures given at the University of Michigan, 1958, tłum. ros., Moskwa, 1963 [Computer Programming and Artificial Intelligence. An intensive course for practicing scientists and engineering. Lectures given at the University of Michigan, summer 1958, College of Engineering, Engineering Summer Conference.]

[C4] McCracken D.D. A Guide to FORTRAN Programming, John Wiley and Sons, 1964

[D1] Dańda J., Ryżko J. Tendencje rozwojowe organizacji pamięci maszyn cyfrowych, [w:] Problemy przetwarzania informacji, t.1, WNT, 1970

[D2] Desmonde W.H. Computers and their uses, Prentice Hall 1964; wyd. polskie: Maszyny matematyczne i ich zastosowania, PWN, 1969

[D3] Dellert G.T. Jr A Use of Macros in Translation of Symbolic Assembly Language of One Computer to Another *Comm. ACM* 1965 (8) 12, s.742-748

[D4] Dańda J. Dziś i jutro maszyn cyfrowych, *Maszyny Matematyczne* 3/67

[E1] Empacher A.B. Maszyny liczą same?, Wiedza Powszechna i Sztandar Młodych, 1960

[E2] Emery G. Electronic Data Processing, Pitman, London, 1968

[E3] Edeleman K.E. A short guide to the wonderful world of COBOL, *Datamation* 12/69, s.161-164

[E4] Empacher J., Maroński J., Sadowski A., Tarasiuk J. Autokod ALGOL dla maszyny URAL-2, PWN, Warszawa, 1966

[E5] Ewreinow E.W., Kosariow J.G. Odnorodnyje uniwersalnyje wyczislitielnyje sistiemy wysokoj proizwoditielnosti, Nowosibirsk, 1966

[F1] Fisher F.P., Swindle G.F. Computer Programming Systems, New York 1964; tłum. ros. Sistiemy programmirowanija, Moskwa, 1971

[G1] Greniewski H. Elementy logiki formalnej, PWN, 1955

[G2] Golden J.T. FORTRAN IV – Programming and Computing, Prentice Hall, 1965

[G3] Gabrini P. Étude d'un systeme de programmation en commande numérique, *Automatisme* 4/70 s.176-180

[G4] Gaines R. Stockton R. On the Translation of Machine Language Programs, *Comm. ACM* 1965, 8 No 12, s.736-741

[G5] Głowacki B. Współczesne systemy cyfrowe, *ETO Nowości* 2/69 s.3-23

[G6] Glauthier Computer Time Sharing: its Origins and Development, *Computers and Automation* 10/67 s.23-28

[H1] Higman B. A Comparative Study of Programming Languages, McDonald, London, 1968

[H2] Hellerman H. Digital Computer System Principles, McGraw Hill Book Co, 1967

[H3] Hellwig Z. (red.) O maszynach cyfrowych, PWE, Warszawa, 1970

[H4] Hicks H.T. Jr Modular Programming in COBOL, *Datamation* 5/68

[H5] Humby E. ICT COBOL Rapidwrite, w: Introduction to System Programming (P.Wegner, ed), Academic Press, 1964

[H6] Halpern M.I. Machine Independence: its Technology and Economies, *Comm. ACM* 12/65, s.782-785

[H7] Hicks H.T. Jr The Air Force COBOL Compiler Validation System, *Datamation* 8/69, s.73-74, 76-77, 81

[H8] Hicks H.T. Jr A Communication Facility for COBOL *Datamation* 12/69, s.148, 153-158

[H9] Hicks H.T. Jr ANSI COBOL, *Datamation* 1.11.1970

[I1] Ivoll T.E. Electronic Computers; tłum. ros. Maszgiz, Moskwa, 1959

[J1] Junkier J.P., Boward G.R. COBOL vs FORTRAN. A Sequel, *Datamation* 4/65, s.65-67

[J2] Jerzykiewicz K., Szczepkowicz J. ALGOL 1204, poz. 1204-VIII-2, ELWRO, 1970

[K1] Kitow A. I. Programmirowanije informacionno-logiczeskich zadacz, Sowietskoje Radio, Moskwa, 1967

[K2] Klepacz W. Pamięci masowe maszyn cyfrowych, WNT, Warszawa, 1970

[K3] Korolew M.A. Obrabotka ekonomiczeskoj informacji na elektronnych maszinach, *Ekonomika*, 1964

[K4] Kriebel C.H. The evaluation of management Information Systems, *IAG Journal* vol.4 No 1, 1971, s.1-14

[K5] Klepacz W. Zastosowanie maszyn matematycznych dla automatyzacji zarządzania, WNT, Warszawa, 1965

[L1] Liapunow A.A. O logiczeskich schemach programm, *Woprosy kibernetyki* 1/58, 8.46-127

[L2] Lippit A. COBOL and Compatibility, *Comm. ACM* 5/62, s.254-255

[L3] Ledley R.S. Digital Computer and Control Engineering, McGraw Hill, 1960

[L4] Lombardi L.A. Mathematical Structure of Nonarithmetic Data Processing Procedures, *Journal of the ACM* 1962, v.9, 1, s.136-159

[L5] Łukaszewicz L. Automatyzacja programowania w Polsce do roku 1970, *Informatyka* 3/71

[L6] Łukaszewicz L. EOL – język do przetwarzania symboli, *Maszyny Matematyczne* 5/69, s.8-11

[M1] Małuszyński J., Witaszek J. Wiedeńska metoda opisu języków programowania, *Informatyka* 3/71

[M2] Mazurkiewicz A. Problemy języków formalnych w automatycznym przetwarzaniu informacji, [w:] Problemy przetwarzania informacji, t.1, WNT, 1970

[M3] Moyle M.P. Introduction to Computers for Engineers, John Wiley, 1967

[O1] Oswald H. Translation by XACT, *Datamation* 1/67, s.37-38

[O2] Opler A. Fourth generation software, *Datamation* 1/67, s.22-24

[O3] Opler A. The receding future, *Datamation* 9/67, s.31-32

[P1] Pawlak Z. Organizacja maszyn bezadresowych, PWN, 1965

[P2] Pawlak Z. Sygnały, symbole, maszyny, Wiedza Powszechna, 1965

[P3] PL/I Language. Specyfication, IBM, 1966

[P4] Paszkowski S. Język ALGOL 60, PWN, 1965

[R1] Raymond F.H. L'automatique des informations, Masson et C. Editeurs, Paris, 1957

[R2] Ruderman M.E. Fappalardo A.N. The hospital computer comes of age, *Computers and Automation* 6/70, s.29-32

[R3] Rubey R.J. A comparative evaluation of PL/I, *Datamation* 12/68, s.21-38

[R4] Revised Report on the Algorithmic Language ALGOL 60, IFIP, 1962 pod red. P. Naura

[R5] Radin G. Rogoway H.P. NPL – new programming Language, 8.1*Comm. ACM* 1965 (8) 1

[R6] Rottmann H.H. Programmieren leicht gemacht mit RPG IBM 360/20, Carl Hauser Verlag, Munchen, 1968

[R7] Reynolds C.H. Software development and its costs, *Computers and Automation* 2/67, s.18-21

[R8] Rishel W.J. Incremental compilers, *Datamation* 1/70, s.129-136

[R9] Ryznar Z. Sterowanie procesami technologicznymi za pomocą elektronicznych maszyn cyfrowych, *Organizacja-Samorząd-Zarządzanie* 4/67, s.204-208

[R10] Ryznar Z. Komputery w społeczeństwie XXI wieku, *Problemy* 8/69, s.483-486

[R11] Ryznar Z. Zarządzanie w czasie rzeczywistym, *Organizacja-Samorząd-Zarządzanie* 9/66, s.516-519

[R12] Richards R.K. Digital computer components and circuits, D. van Nostrand Co, 1958

[S1] Sammet E.J. Programming languages: History and fundamentals, Prentice Hall, 1969

[S2] Sammet E.J. Fundamental concepts of programming languages, *Computers and Automation* 2/67 s.30-31,34-35

[S3] Shirley D. Comparing COBOL's, *Data and Control Systems* 1/67

[S4] Szuprowicz B.O. The time-sharing users: who are they?, *Datamation* 8/69

[T1] Tatarkiewicz Władysław Historia filozofii, t.III, PWN, 1958

[T2] Targowski A. Automatyzacja przetwarzania danych, PWE, 1970

[T3] Trachtenbrot B.A. Algorytmy i maszinnoje reszenije zadacz, Moskwa, 1960

[T4] Teague R.G., Brady A.H. COAX: a Preprocessor for COBOL, *Datamation* 7/69
[V1] Vocabulary of Information Processing, IFIP-ICC, North Holland, 1968
[W1] Wilkes M.V. Computers then and now, *Journal of the Association for Computing Machinery* 1/68, s.2-7
[W2] Warmus M. GIER-ALGOL, PWN, W-wa, 1966
[W3] Wilkes M.V. Time-sharing computer systems, Mc Donald, London, 1970
[X1] *Datamation* 7/69, s.69,79
[X2] *Computer Weekly* 14.1.71
[X3] *Datamation* 4/69, s.199
[X4] *Datamation* 6/67, s.31
[X5] *Data Processing* 9-10/67, s.17-20
[X6] *Computers and Automation* 12/68, s.23
[X7] *Nowości ETO* 2/69
[X8] *Data and Control Systems* 1/67
[X9] *Sowriemiennoje programmirowanije*, zbiór art. tłum. z ang., Moskwa, 1966
[X10] *Computers and Automation* 5/70, s.77
[X11] *Comm. ACM* 1965 (8) 1, 5
[X12] *Datamation* 2/69, s.ll
[X13] *Introduction to IBM data processing systems*, IBM, 1969
[X14] *COBOL ICL System 4-50 Manual*
[X15] *COBOL ICL Series 1900 Manual*
[X16] *Maszyny Matematyczne* 2/67
[X17] The next generation from 50 viewpoints, *Datamation* 1/67, s.31-34
[X18] Burroughs chosen to build ILLIAC IV, *Computers and Automation* 4/67, s.51
[X19] *Datamation* 7/69, s.203
[X20] *Maszyny Matematyczne* 5/68, s.21-22
[X21] *Elektronische Datenverarbeitung* 2/70

DODATEK 1

Tabela 200 faktów i więcej (vf3a) **© dr inż. Zygmunt Ryznar**

ŚWIATOWE KALENDARIUM INFORMATYKI DO 1971 r.

over 200 events in computing - worldwide timelines by 1971

Wynalazki techniczne-teorie-hardware-software

Wstęp od autora

Obecna finalna wersja kalendarium, stanowi uzupełnienie do książki "Zarys historii programowania elektronicznych maszyn cyfrowych" z 1972 r. Uzupełnienie było konieczne ze względu na ujawnienie nowych źródeł informacj oraz upływ czasu, dzięki któremu można bardziej obiektywnie i syntetycznie ogarnąć to, co się wydarzyło w ciągu kilkudziesięciu minionych lat.

Do zestawienia weszły wydarzenia pionierskie techniczne (np. wynalezienie tranzystora, układu scalonego, pamięci typu X), teoretyczne (np. zapis Backusa, notacja beznawiasowa), językowe (pierwsze języki programowania w danej klasie), pojawienie się pierwszych komputerów danej klasy czy generacji, software'owe (systemy operacyjne, systemy zarządzania bazą danych itp.). Tabela zawiera również wydarzenia informacyjne (np. utworzenie znaczących firm komputerowych), których nie można pominąć pisząc o historii informatyki. Umieszczono również kilka informacji podsumowujących (np. liczba języków programowania w roku x). Pozycje te posiadają w kol. "kraj" znak * i nie są zliczane do liczby zdarzeń wynalazczych.

Ze względu na ograniczenie horyzontu czasowego do 1971 roku do zestawienia nie mogły wejść bardziej współczesne technologie jak np. pochodne języka C, hurtownie danych, superkomputery o mocy liczonej w tf, internet, systemy operacyjne Windows, Linux itp. Niekiedy omawiając wydarzenie podaję informacje z nią związane wykraczające poza 1971 rok (np. pisząc o organizacji CODASYL nie mogłem nie wspomnieć o polskim systemie zarządzania bazą danych RODAN, gdyż być może było to pierwsze i zarazem ostatnie nasze znaczące dokonanie w zakresie oprogramowania narzędziowego).

Fakty zebrane w tabeli pochodzą głównie z moich archiwów, uzupełnionych internetowymi tekstami źródłowymi (tutaj pomocna okazała się lista dyskusyjna Sekcji Historycznej PTI). Starałem się nie korzystać z nieudokumentowanych publikacji typu "timelines". Wybierałem z wydarzeń (opisywanych zwykle w książkach i obszernych artykułach) tylko to, co uważałem za istotne i ciekawe, z własnej i polskiej perspektywy. Nie interesowały mnie kryteria amerykańskie czy angielskie lub inne, tylko moje własne wypracowane podczas ponad 50 lat pracy w charakterze informatyka.

Poczatkowo dążeniem moim było, aby tabela była zwięzła (maksymalnie 2-3 wiersze na pozycję), pełniąc rolę jakby indeksu wydarzeń. Potem doszedłem do wniosku, że takie "suche" zestawienie faktów będzie mało interesujące i dodałem bardziej szczegółowe informacje (w tym porównawcze, nawiązujące do wcześniejszych lub późniejszych wydarzeń), których nie ma w mojej wyżej wspomnianej książce .

Nie dołączam źródeł bibliograficznych, gdyż byłoby to zestawienie kilku tysięcy pozycji, kwalifikujące się do odrębnego opracowania.

Jeszcze jedna uwaga. Tabela jest wielotematyczna i było to zamierzenie umyślne. Zapewne zestawienia poświęcone odrębnym działom (np. językom programowania, wynalazkom technicznym itp.) są też godne uwagi, ale powinny występować w przypadku bardziej rozbudowanych analitycznych materiałów, skierowanych do określonej grupy specjalistów. Zwięzła tabela wielotematyczna ma tę zaletę, że pozwala lepiej kojarzyć różne strony tej samej "rzeczy" (czytaj "techniki komputerowej" gdyż tak można byłoby określić informatykę). Jeśli np. powstały pamięci dyskowe to stworzyło to podstawy do nowych rozwiązań w zakresie systemów operacyjnych i języków programowania.

Na koniec uwaga o datach wydarzeń - czasem są one różnie datowane w różnych źródłach. Starałem się wtedy dotrzeć do najwłaściwszego z nich, ale nie zawsze było to możliwe. Ponadto prace badawcze i konstrukcyjne mogą trwać lata - wówczas pojawić się może okres "od-do" albo kilka dat: publikacja raportu o pracy, pierwsze uruchomienie, wejście na rynek komercyjny itp.

Pod uwagę należy wziąć również "preferencje" narodowościowe autorów źródłowych informacji, a więc skłonność do przyznawania pierwszeństwa w datach dla faktów pochodzących z ich krajów.

Podsumowując, wydaje się, że okres dwudziestolecia 1951-1971 był najwazniejszy, gdyż stworzył podwaliny epoki komputerowej od strony teoretycznej i technicznej. Po roku 1980 poczyniono następne najistotniejsze kroki milowe - pojawienie się komputerów osobistych i systemów Windows, technologii "data-mining" i hurtowni danych, światowej sieci komputerowej użytkowanej poprzez internet, bezpłatnych systemów operacyjnych (np. Linuxa) i bezpłatnego oprogramowania uzytkowego na licencji GNU. Komputery mogły więc zawędrować pod każdą "strzechę".

Zygmunt Ryznar

Statystyka tabeli faktów do 1971r.

wg stanu wpisów na dzień 28 lutego 2011

Liczba pozycji 204
w tym zdarzenia wynalazcze:

- USA 96
- UK 18
- Niemcy 10
- Polska 7 (w sumie 17 pozycji, ale 8 można uznać za wynalazki)
    1. Brunon Abakanowicz - integraf

2. Abraham Stern - arytmometr ręczny, "machina pierwiastkująca" i "machinę rachunkowa"
3. Zelig Slonimski - "instrumenty" liczące,
4. Jan Łukasiewicz łącznie za osiągnięcia: -notacja polska i trójwymiarowa logika
5. Z.Pawlak - koncepcja maszyn bezadresowych
6. złamanie kodu enigmy przez polskich matematyków
7. Stanisław Ulam - metoda Monte Carlo
8. system zarządzania bazą danych RODAN [1979-1980]

- Holandia 5
- ZSRR 4
- Japonia 3
- Szwajcaria 3
- Francja 3
- Norwegia 2
- Australia 2
- Szwecja 1

Fakty mówią same za siebie, ale pozwalam sobie zauważyć zjawisko "ostrej" ("łeb w łeb") rywalizacji USA i Anglii (UK) w latach 50-tych, zarówno na płaszczyźnie konstrukcyjnej jak i teoretycznej.

Data	Wyszczególnienie	Kraj
_500 p.n.e.	Panini [Paniani] opracowuje notację Classical Sanskrit standaryzującą gramatykę i morfologię Sanskryptu. Podobno do tej notacji zbliżony jest współczesny zapis BNF (Backus Naur Form)	[Persja]
_500 p.n.e.	Hinduski matematyk Pingala po raz pierwszy opisuje binarny system liczenia	[Indie]
_300 p.n.e.	W "Elementach" Euclidesa opisane są nietrywialne algorytmy (np. nawiększy wspólny dzielnik)	[Grecja]
_300 p.n.e.	Abacus - liczydło (deska z wyżłobionymi rowkami, które symbolizowały kolejne potęgi dziesięciu)	[Babilonia]
80 n.e.	Antikythera - urządzenie mechaniczne z brązu do obliczeń gwiezdnego (lunar) kalendarza ze zmienną liczbą miesięcy w roku	[Grecja]

820-825	Abu Abdallah Muhammad ibn Musa al-Chorezmi [al-Khwarizmi] czyli "Muhammad syn Musy z Chorezmu", opisał w pracy "On the Calculation with Hindu Numerals" pozycyjny (dziesiętny)system liczbowy. Uważa się, że od "al-Khwarizmi" pochodzą terminy "algorytm" i "algebra".	[Azja]
1500	Leonardo da Vinci wykonuje szkic prostego kalkulatora mechanicznego	[Włochy]
1600	John Napier buduje tabliczkę do mnożenia zwaną kostkami Napiera	[Szkocja]
1621	William Oughtred tworzy suwak logarytmiczny. Ciekawostka: jako pierwszy użył on znak X jako symbol mnożenia w pracy "Clavis Mathematicae".	[UK]
1623	Wilhelm Schickard konstruuje pierwszy mechaniczny zegar-kalkulator	[Niemcy]
1642	Blaise Pascal w wieku 19 lat -aby pomóc ojcu, który był poborcą podatkowym- buduje sumator mechaniczny, zwany maszyną arytmetyczną, mogący dodawać i odejmować	[Francja]
1671	Gottfried von Leibniz tworzy maszynę liczącą, która wykonuje działania dodawania, odejmowania, mnożenia i dzielenia. Opublikował pracę, w której omawia binarny system liczenia.	[Niemcy]
1777	Charles Stanhope konstruuje mechaniczną maszynę liczącą	[UK]
1801	Joseph-Marie Jacquard opracowuje sposób sterowania krosnami za pomocą kart perforowanych	Francja
1812-1817	Abraham Stern konstruuje arytmometr ręczny, "machinę pierwiastkującą" i "machinę rachunkową"	[Polska]
1822	Charles Babbage rozpoczyna budowę mechanicznej maszyny rachunkowej, tzw. maszyny różnicowej. Projekt zrealizowany dopiero w 1992 roku, znajduje się w muzeum techniki w Science Museum, Londyn	[UK]
1833	Projekty "maszyny analitycznej" Charlesa Babbage	[UK]
1843-1845	Zelig Slonimski tworzy "instrumenty" liczące (dodawanie, odejmowanie, mnożenie) w Petersburgu	[Polska]
1847	George Boole publikuje pierwsze prace w dziedzinie logiki symbolicznej. W 1854 roku sformułował system logiczny (nazwany potem algebrą boole'owską), który stanowił wsparcie binarnego systemu liczenia	[UK]
1878	Brunon Abakanowicz tworzy pierwszy działający model integrafu - urzadzenia do całkowania graficznego (następcy planimetru do pomiaru powierzchni).Patent uzyskuje w 1880r i odtąd jego urządzenie było produkowane przez szwajcarską firmę Coradi. W tym samym czasie podobne urządzenie buduje Wawrzyniec Żmurko.	[Polska]
1882	Powstaje **Ferranti** (Sebastian Ziani de Ferranti)	[UK]*
1884	John H. Patterson zakłada firmę NCR - National Cash Register Company	[USA]*
1888	William S. Burroughs występuje o patent na mechaniczny sumator.	[USA]
1893	Utworzono **TMC** Tabulating Machine Corporation (Herman Hollerith)	[USA]*
1903	Nicola Tesla, fizyk pochodzenia serbsko-chorwackiego, patentuje elektryczne bramki logiczne.	[USA]
1904	John A. Fleming wynajduje diodę lampową i konstruuje prostownik na bazie tych elementów	[UK]
1907	Lee de Forest wynajduje triodę	[USA]

1910	Namihei Odaira tworzy firmę Hitachi	[Japonia]*
1911	Thomas Watson Sr tworzy firmę CTR (Computing Tabulating Recording Corporation), która potem zmieni nazwę na IBM	[USA]*
1917	Jan Łukasiewicz wprowadza trójwartościową (tak,nie,niewiadoma) logikę (three-valued propositional calculus)	[Polska]
1917	W Kanadzie CTR zmienia nazwę na IBM	[Kanada]*
1919	Wynalezienie przerzutnika (trigger,flip-flop) (William Eccles i F. W. Jordan, później Otto Herbert Schmitt).	[USA]
1922	Powstaje organizacja GAMM (Gesellschaft für Angewandte Mathematik und Mechanik). Na polu informatycznym wsławiła się zasługą (wspólnie z ACM) wsparcia opracowania ALGOLu 58	[Niemcy]*
1924	Jan Łukasiewicz wprowadza beznawiasowy przedrostkowy (prefiksowy) zapis wyrażeń - notację polską -popularnie zwany polskim zapisem. Zastosowanie praktyczne w komputerach znalazł zapis postfiksowy czyli Odwrotna Notacja Polska (ONP) - Reverse Polish Notation (RPN), którą zaproponowali w 1954 r Burks, Warren i Wrightby, algorytmicznie opracował Ch.Hamblin, a potem niezależnie zdefiniowali F. L. Bauer i E. W. Dijkstra. Notacja ONP stała się podstawą budowy rewersyjnych stosów pamięciowych w komputerach (1960 - Burroughs B5000, 1963 - English Electric KDF9). W LISP2 zmodernizowano nieco tę notację i została ona nazwana Cambridge Polish Notation.	[Polska]
1924	W USA CTR zmienia nazwę na IBM	[USA]*
1925	Vannevar Bush konstruuje analogowy komputer do rozwiązywania równań różniczkowych. Kopia jego powstaje w 1930r w MIT i będzie używana do obliczeń toru pocisków artyleryjskich podczas II Wojny Światowej	[USA]
1927	Bernard D.H. Tellegen wynajduje lampę z trzema siatkami czyli pentodę	[Holandia]
1928	Rosyjski imigrant V. Zworykin wynajduje kineskop	[USA]
1928	Fritz Pfleumer patentuje taśmę magnetyczną	[Niemcy]
1932	Polscy matematycy Jerzy Różycki, Henryk Zygalski i Marian Rejewski łamią kod niemieckiej Enigmy.	[Polska]
1936	Konrad Zuse zgłasza patent mechanicznego komputera pracującego w arytmetyce zero jedynkowej czyli binarnej. Zuse jest też pomysłodawcą idei operacji zmiennoprzecinkowych oraz arytmometru ósemkowego.	[Niemcy]
1936	John von Neumann - niezależnie od Francuza L.Couffignala proponuje użycie dwójkowego systemu liczenia w maszynach liczących	[USA]
1937	Georg R. Stibitz z Bell Labs tworzy prosty cyfrowy kalkulator na bazie przekaźników telekomunikacyjnych.	[USA]
1937	Praca C.Shannona n/t budowy układów cyfrowych w oparciu o algebrę boole'wską i system binarny	[USA]
1937	Alan Turing opisuje w publikacji "On computable numbers" teoretyczne podstawy procesu obliczania (maszynę Turinga), twórca teorii automatów	[UK]
1938	Konrad Zuse kończy prace nad pierwszym mechanicznym komputerem cyfrowym -ZI, operujący na binarnym systemie liczb i stosujący rachunek zmiennopozycyjny.	[Niemcy]
1938	David Packard i William Hewlett tworzą firmę HP (Hewlett Packard)	[USA]*

1941	Konrad Zuse kończy prace nad pierwszym programowalnym komputerem cyfrowym, zbudowanym na przekaźnikach, nazwanym Z3	[Niemcy]
1942	John Atanasoff i Clifford Berry budują komputer **ABC** (**A**tanasoff-**B**erry **C**omputer) w Iowa Univ. w latach 1937-42 (arytmetyka dwójkowa, zastosowano lampy próżniowe, kondensatorowe pamięci)	[USA]
1943	Konrad Zuse tworzy kalkulator Z3 działający na 22 cyfrowych liczbach dwójkowych	[Niemcy]
1943	Paul Eisler patentuje płytkę drukowaną	[Niemcy]
1943	Alan Turing i Thomas Flowers wraz z zespołem tworzą elektroniczny lampowy komputer deszyfrujący (do łamania kodu Enigmy)- Colossus, który był następcą maszyny deszyfrującej "Bombe". Algorytmy na których pracował Bombe (oraz prawdopodobnie w dużej mierze konstrukcja) oparte były na dokumentacji opracowanej przez polskich matematyków Jerzego Różyckiego, Henryka Zygalskiego i Mariana Rejewskiego, którą przywiózł do Anglii z Polski A. Turing. "Bombe" (bomba kryptologiczna) nosiła nazwę "Heath Robinson" i została zbudowana przez Maxa Newmana i Wynn-Williamsa niedługo przed Colossusem, którego eksploatację rozpoczęto w czerwcu 1944 r.	[UK]
1944-1945	Matematyk węgierskiego pochodzenia John von Neumann definiuje architekturę programowalnego komputera, w której główną ideą było przechowywanie programu w pamięci	[USA]
1944	Howard Aiken wraz z zespołem buduje elektromechaniczny komputer nazwany Harward Mark 1, składający się z 800 tys. elementów elektromechanicznych	[USA]
1945	Konrad Zuse opracowuje język programowania wysokiego poziomu, zwany Plankalkûl - w którym napisano program do figur szachowych (a więc do gry w szachy)	[Niemcy]
1946	J. W. Mauchly, J. P. Eckert wraz z zespołem kończą prace nad uniwersalnym komputerem elektronicznym ENIAC. Zakończył swą służbę 2 października 1955 roku	[USA]
1946	Stanisław Ulam dla Los Alamos Scientific Laboratory opracowuje metodę obliczeń procesów statystycznych na podstawie prób losowych znaną jako metoda Monte Carlo - ma ona przyspieszyć obliczenia nad bronią jądrową.	[USA-Polska]
1946	Powstaje IEEE (Institute of Electrical and Electronics Engineers)	[USA]*
1946	Freddie Williams składa wniosek patentowy na pamięć opartą na lampie oscyloskopowej (cathode ray tube-CRT) o pojemności 512-1024 bitów.	[USA]
1947	IBM buduje lampowo-przekaźnikowy kalkulator SSEC (Selective Sequence Electronic Calculator), który zapowiada odchodzenie z epoki maszyn licząco-analitycznych	[USA]
1947	Frederick Viehe patentuje wynalazek pamięci ferrytowej, skonstruowanej w domowym laboratorium. Niezaleznie od niego pamięć ferrytową tworzą niedługo później Amerykanie chińskiego pochodzenia An Wang and Way-Dong Woo.	[USA]
1947	W. Shockley, W. Brattain i J.Bardeen tworzą pierwszy germanowy tranzystor ostrzowy	[USA]

1947	Operatorzy komputera przekaźnikowego Mark II jako przyczynę wadliwego działania wskazują pluskwę(bug) w jednym z przekaźników. Grace Murray Hopper spopularyzowała to wydarzenie jako "odpluskwianie" (debugging) programu. [*ten fakt jest tylko ciekawostką i nie liczy się w statystyce wynalazków*]	[USA]*
1947	Powstaje międzynarodowa organizacja ACM (Association for Computing Machinery)	[USA]*
1947	Powstaje ISO (Międzynarodowa Organizacja ds. Standardów)	[ ]*
1947-1952	Murray Grace Hopper opracowuje assembler. Pełniejsze kompilatory (A-0, A-1) tworzy w 1955 r w Remington Rand dla komputerów UNIVAC	[USA]
1947	Rozpoczęcie stosowania bębnów magnetycznych w Komputerach (MARKIII, ERA1101, itp.)	
1948	C. Shannon tworzy teoretyczne podstawy telekomunikacji cyfrowej	[USA]
1948	W Manchester University powstaje komputer "Baby" z pamiętanym programem. 1951 r.- rozpoczęcie produkcji tych komputerów przez firmę Ferranti jako Ferranti Mark 1	[UK]
1948	W komputerze ENIAC zastosowano pamięć typu ROM do przechowywania Function Tables	[USA]
1949	Pojawia się - zaproponowany przez J.Mauchly'ego, współtwórcę ENIACa- pierwszy "przenaszalny" (stosowalny nie tylko na jednej maszynie) język programowania Short Code (na komputerach BINAC i UNIVAC). Był tak prosty (ale zawierał zmienny przecinek), że mozna było go ręcznie "przekompilować" na kod maszynowy	[USA]
1949	Francis Holberton tworzy generator programów (dla UNIVAC), który generuje m.i. programy sort i merge	[USA]
1949	Pierwsze lampowe komputery przechowujące program w pamięci operacyjnej: -EDSAC(1) zespół Maurice'a Wilkes'a na Univ.Cambridge (UK), -SSEC zespół Johna Prespera Eckerta w N.Jorku. Programy mogły zawierać skoki warunkowe. Niezależnie od Neumanna, Wilkes wskazał, że należy dokonywać operacji arytmetycznych na adresach, osiągając przez to znaczne skrócenie wielkości programu (jak wiadomo, modyfikacja adresów pozwala na tworzenie tzw. pętli w programie). Docenił też walory stosowania biblioteki podprogramów.	[UK-USA]
1950-1951	Maurice Wilkes (Cambridge Computer Laboratory) tworzy koncepcję mikroprogramu przechowywanego w pamięci ROM. Wdrożono ją kilka lat potem (1958?) w komputerze EDSAC(2) (Manchester Univ.), który posiadał moduł o nazwie "Microprogram Control Unit". Dodać trzeba, że pod koniec lat 50-tych mikroprogramowe implementacje wystąpiły również w USA.	[UK]
1950	Powstaje komputer lampowy CSIRAC (Council for Scientific and Industrial Research Automatic Computer, na którym w latach 1950-1951 odtwarzano muzykę. Ten pierwszy australijski komputer nie był efemerydą, gdyż pracował do 1964 r.	[Australia]
1950	Jay Forrester wynajduje pamięć ferrytową	[USA]
1950	Po raz pierwszy zastosowano bęben magnetyczny w komputerze UNIVAC/ERA 1101 w U.S. Navy	[USA]

1950	Powstaje komputer ACE zbudowany przy udziale A.Turinga	[UK]
1950	Na Imperial University Tokyo Yoshiro Nakamats konstruuje floppy dysk. Licencję sprzedaży dyskietki przejmuje IBM	[Japonia]
1950	Powstaje jeden z pierwszych w Europie programowalny lampowy komputer MESM (Małaja Elektronno-Scziotnaja Maszyna)	[ZSRR]
1951	Firma Jacob Instrument stosuje pamięć ferrytową w komputerze Jain Comp	[USA]
1951	Allec Glennie opracowuje kompilator AUTOCODE (dla Ferranti Mark1) - pierwszy kompilator ogólnego użytku	[UK]
1951	Powstają pierwsze komputery do zastosowań biznesowych (UNIVAC i LEO)- dotychczasowe komputery służyły do obliczeń naukowo-technicznych, a nie do masowego przetwarzania danych ekonomiczno-administracyjnych	[USA,UK]
1952	Pod kierownictwem S.A.Lebiediewa powstaje lampowy komputer BESM-1 (Balszaja Elektronno-Scziotnaja Maszyna) jeden z najszybszych wówczas komputerów w Europie 8–10 KFlops	[ZSRR]
1952	Heinz Rutishauser ze Szwajcarskiego Instytutu Technologicznego proponuje naturalną notację dla wyrażeń matematycznych [obejmującą też m.i. typowy dzisiaj zapis pętli for k=1 (1) 10] jako wejście do komputera i tworzy koncepcję kompilatora takiego zapisu.	[Szwajcaria]
1952	J.H.Laning i W.Zierler tworzą w MIT dla komputera Whirlwind pierwszy w USA interpreter matematycznych wyrażeń pisanych "naturalnie"	[USA]
1953	W IBM zespół J.Backusa i J.Sheldona tworzy symboliczny interpretacyjny Speedcoding System dla IBM701, którego cechą charakterystyczną były dwa różnoadresowe zbiory operacji (trzyadresowy i jednoadresowy)	[USA]
1952-1953	A.A.Liapunow opracowuje tzw. operatorową notację algorytmów.	[ZSRR]
1952	IBM dostarcza na rynek swój pierwszy komercyjny komputer IBM 701, wyposażony w pamięć na lampach oscyloskopowych (2045-4096 słów) oraz bęben magnetyczny o pojemności 8192 słów. Niedługo potem dostaje pamięć na taśmach magnetycznych.	[USA]
1953	W Manchester Univ. uruchomiono prototyp prawdopodobnie 1szego w świecie tranzystorowego komputera - pełna wersja eksploatacyjna powstała w 1955 r. W tymże 1955 roku IBM demonstruje pierwszy kalkulator oparty na tranzystorach.	[UK]
1953	IBM tworzy systemy operacyjne SOS (Share Operating System) i FMS (Fortran Monitor System) dla IBM709 i 704	[USA]
1954	Pierwsza symulacja sieci neuronowej na komputerze w MIT przez Farleya i Clarka - inspirowana przez prace badawcze (nad regułami uczenia) z 1949 r Donalda Hebba z McGill University w Kanadzie.	[USA-Kanada]
1954	w Texas Instruments utworzono pierwszy tranzystor krzemowy	[USA]
1954	Użycie taśmy magnetycznej w komputerze UNIVAC I (podobno 1sze użycie taśmy magnetycznej nastąpiło w 1949 r w komputerze BINAC)	[USA]
1955	IBM informuje o stworzeniu dysku magnetycznego 5MB (RAMAC 305)-we wrześniu 1956 roku sprzedaje 1szy komputer z tym dyskiem	[USA]

1955	W IBM powstaje komputer IBM 705 wyposażony w ferrytową pamięć rdzeniową (20 tys. znaków) i dysk magnetyczny (60 tys. znaków). Rok później pamięć ferrytową instaluje Remington Rand w maszynie UNIVAC 1103A.	[USA]
1956	Przekształcenie GAM (Grupa Aparatów Matematycznych) w ZAM (Zakład Aparatów Matematycznych)	[Polska]*
1956	Noam Chomsky publikuje "Three models for the description of language" - zawiera 3 stopniowy model gramatyk formalnych zwany "Chomsky Hierarchy", który łączy teorię obliczeń i języków formalnych	[USA]
1956	W Lincoln Laboratories MIT zbudowany zostaje pierwszy komputer tranzystorowy szerszego przeznaczenia.	[USA]
1956-1957	John Backus wraz z zespołem z IBM wprowadzają do użycia FORTRAN (FORmula TRANslating System) - pierwszy język wysokiego poziomu powszechnego użycia	[USA]
1956-1958	John McCarthy tworzy język programowania LISP, nadający się do wykorzystania w dziedzinie sztucznej inteligencji. W LISP zastosowano struktury listowe i rekursję funkcji z użyciem zmodyfikowanej polskiej notacji.	[USA]
1956-1958	IPL (Information Processing Language)- język do struktur listowych opracowany przez Allena Newella,J.C.Shawa i Herberta Simona. Jednym z programów napisanych przez nich w tym języku w 1958 r był program do gry w szachy. IPL-V implementowano w latach 60-tych na wielu komputerach (najpierw na IBM 650 w 1958 r).	[USA]
1956	W Remington Rand powstaje komputer UNIVAC 1103A (ERA 1103) wyposażony w rdzeniową pamięc ferrytową - był następcą komputera UNIVAC 1103 zbud. w 1953 z pamięcią na lampach oscyloskopowych	[USA]
1957	W Remington Rand dla UNIVAC I powstaje "konkurencyjny" w stosunku do Fortranu język MATH-MATIC (AT-3), w którym po raz pierwszy zastosowano automatyczną segmentację programu, który okazywał się za duży (w dzisiejszym języku "overlay'e" czyli nakładki)	[USA]
1957	Alex Bernstein tworzy (prawdopodobnie pierwszy w świecie - nie licząc prób Zusego w języku Plankalkûl) program do gry w szachy na IBM 704. Na 1 ruch komputer potrzebował ok.8 min. Pierwszy użytkowy program do gry w szachy o nazwie MacHack VI napisał Richard Greenblatt w latach 60-tych w MIT. Program ten mógł brać udział w turniejach szachowych z udziałem ludzi, korzystał z "ksiąg posunięć" (book) itp.	[USA]
1957	Wynalazek kriotronu (przewidywano jego uzycie w charakterze pamięci) - planarny kriotron tworzy Dudley Allen Buck w Massachusetts Institute of Technology	USA
1957	Ken Olsen tworzy firmę Digital - DEC, która zapoczątkowuje erę minikomputerów	[USA]
1957	Willami Norris i Seymour Cray tworzą firmę CDC	[USA]*
1957	Charles Hamblin opracowuje algorytm Odwrotnej Polskiej Notacji i buduje pierwszy stos w New South Wales University of Technology.	[Australia]

1957	Profesor MIT Noam Chomsky publikuje "Syntactic Structures", w której podaje teorie transformatywnych gramatyk geneeratywnych.	[USA]
1957	W komputerze NCR304 zastosowano mikroprogram do realizacji autokodu.	[USA]
1958	Publikacja monografii A.P. Jerszowa "Programmirujuszczaja programma dla bystrodieistwujuszczej elektronnoj scziotnoj masziny" - jedna z pierwszych prac badawczych w dziedzinie automatyzacji programowania	[ZSRR]
1958	Jack Kilby z Texas Instruments tworzy działający układ kilku elementów na jednym podłożu półprzewodnikowym	[USA]
1958	Powstaje modem AT&T Bell 101 , który umożliwia transmisję danych komputerowych z szybkością 110 baudów przez sieć telefoniczną w trybie wybierania numeru (dial-up)	[USA]
1958	ALGOL. W 1958 r. powstaje język IAL (International Algebraic Language) - czyli ALGOL 58 (następnie Algol 60, Algol 68) - Był to pierwszy język opracowany przez międzynarodową grupę ekspertów z Europy (GAMM) i USA (ACM). Język ten zyskał uznanie głównie wśród naukowców w Europie. Niestety - zapewne wskutek braku wsparcia implementacyjnego ze strony amerykańskich producentów komputerów, nadających ton światowej informatyce - język ALGOL stopniowo zanikał. Pod wpływem ALGOLu powstało kilka języków m.i. LISP, JOVIAL, SIMULA, AED (oprac. w MIT - w nim oprogramowano pierwszą wielowymiarową bazę danych Express firmy IRI).	[Europa-USA]
1958	W GAM (potem IMM - Instytut Maszyn Matematycznych) powstaje komputer XYZ	
1959	Powstaje notacja BNF (Backus Naur Form)	[USA]
1959	Powstaje konsortium CODASYL ("Conference on Data Systems Languages"), którego zadaniem był nadzór i stymulowanie rozwoju standardów języków programowania w celu zapewnienia ich stosowania na wielu komputerach. Organizacja zajmowała się głównie językiem COBOL oraz standardem sieciowych i hierarchicznych baz danych ze strukturami łańcuchowymi (języki DDL - Data Description Language, DML - Data Manipulation Language, schematy i podschematy). Pojawienie się relacyjnych baz danych i ich popularność doprowadziły do zaniku działalności tej organizacji. Wg standardów "codasylowskich" opracowano co najmniej kilka szeroko stosowanych systemów zarządzania bazami danych, a w Polsce w latach 1979-1980 opracowano RODAN pod kierunkiem W.Staniszkisa i A.Dutkowskiego. Podążając zgodnie z rozwojem swiatowej technologii, w Rodanie utworzono moduł komunikacji w języku SEQUEL, który dokonywał "mapowanie" modelu relacyjnego w zapytaniu na sieciowy bazy danych. System stosowano w kilkunastu przedsiębiorstwach głównie w zakresie TPP (Techniczne Przygotowanie Produkcji)	[USA]*
1959	W Manchester Univ. powstaje pierwszy w swiecie prototyp stronicowanej (paging) pamięci wirtualnej, zastosowany w 1962 r. w komputerze ATLAS	[UK]
1959	Na IBM 704 pojawia się DYNAMO - jeden z pierwszych językow symulacyjnych	[USA]

1959	Robert Noyce wynajduje technikę pozwalającą na tworzenie cienkich aluminiowych połączeń na krzemowej płytce. Jest to krok do stworzenia układów scalonych	[USA]
1959	Ch.Strachey na konferencji UNESCO wygłasza pierwszy referat n/t podziału czasu (time-sharing) i pracy wielodostępnej. Badania w tym zakresie prowadzi potem też J.Mc Carthy.	[USA]
1959	J. Hoerni tworzy technologię planarną do produkcji układów scalonych	[Szwajcaria]
1959	J. Kilby z Texas Instruments (układ na germanie) oraz R. Noyce z Fairchild (układ na krzemie) wytwarzają pierwszy układ scalony	[USA]
1959	Powstaje język COBOL (zainicjowany przez Dep.Obrony, a później prace prowadzone w ramach CODASYL i Shart Range Committee)	[USA]
1959	Powstaje **ICT** (fuzja Powers-Samas i British Tabulating Machine Company)	[UK]*
1959	Hitachi wprowadza na rynek komputery tranzystorowe	[Japonia]
1959-1960	Powstanie międzynarodowej organizacji IFIP (International Federation for Information Processing)	[ ]*
1960	Ukazuje się dokumentacja języka LISP 1, implementowanego później na IBM 704. Wersja LISP 2 zawiera znaczne zmiany. Jako ciekawostkę wymienić można, że zmodyfikowano w nim polską notację Łukasiewicza i nazwano ją Cambridge Polish. LISP uważano za jedyny język wysokiego poziomu w owych czasach, w którym programy jak i dane są takimi samymi strukturami (listami), programy mogą zmieniać siebie (a więc spełniona była tutaj idealnie jedna z zasad architektury J.von Neumanna). Język aplikatywny: wszystkie konstrukcje LISPu są wynikiem zastosowania funkcji do argumentów.Występowała rekurencja.	[USA]
1960	W komputerze KDF9 firmy English Electric po raz pierwszy zastosowano zeroadresowy format instrukcji	**[**[UK]
1960	Honeywell - Executive (do zarzadzania pracą wieloprogramową - do 7 programów)	[USA]
1960	Lata 60-te modele tzw. sieciowych baz danych (wg zaleceń DBTG-CODASYL) i hierarchicznych (IMS firmy IBM)	[ ]*
1960	UMC1 (25 egz. do 1964 r.) Polit.Warszawska (PW), potem prod.seryjnie przez ELWRO	[Polska]
1961	W General Electric w systemie GECOM dla komputera GE-225 pojawia się software'owy procesor tablic decyzyjnych TABSOL. Tablice decyzyjne zostały wprowadzone w latach 1957-1958 jako pomoce do analizy systemu (na etapie projektowania aplikacji) i programowania.	[USA]
1961	Ukazuje się B5000 firmy Burroughs, pierwszy komercyjny komputer z segmentowaną pamięcią wirtualną. Zastosowano w nim też mikroprogramową obsługę polskiej notacji.	[USA]
1961	ODRA1001 ELWRO (od 1963 roku prod. ODRA1003/1013 - w sumie 130 szt.)	[Polska]*
1961	25 kwietnia 1961 urząd patentowy przyznał Robertowi Noyce (współzałozycielowi Fairchild SemiConductors) pierwszy patent na układ scalony. 6 lutego 1959 Jack Kilby z Texas Instrument złożył wniosek	**[**USA]

	patentowy, ale patent został przyznany dopiero 23 czerwca 1964 r.	
1961	Pojawia się opis języka GPSS - jednego z pierwszych językow symulacyjnych, zaimplementowanego po raz pierwszy na IBM 7090	[USA]
1961	Opracowanie systemu operacyjnego z podziałem czasu CTSS (Compatible Time-Sharing System) w MIT	[USA]
1961	IBM tworzy dwumaszynowy system STRETCH (jedna maszyna z magistralą szeregową wykonywała czynności przygotowawcze, a druga -równoległa wykonywała obliczenia podstawowe)	[USA]
1961	Powstaje komputer ZAM2 w Zakładzie Aparatów Matematycznych ZAM (potem IMM)	[Polska]*
1961	Powstaje ECMA (European Computer Manufacturers Association)	[ ]*
1961	IBM wprowadza na rynek dysk 1301 o pojemności 28MB	[USA]
1961-1962	K.E.Iverson w Harvard Univ. opracowuje precyzyjną notację złożonych procesów sekwencyjnych (pośrednią pomiędzy "rozwlekłym" zapisem BNF i "anonimowym" zapisem Chomsky'ego). Notacja ta znana jest też pod nazwą języka APL (Array Processing Language)	[USA]
1962	powstaje ATLAS - komputer o zaawansowanej architekturze (m.i. pamięć wirtualna sprzężona z pamięcią asocjacyjną, szybkość 1ml.op/sek) zbudowany jako wspólne przedsięwzięcie University of Manchester, Ferranti i Plessey . Pracował do 1971r.	[UK]
1962	S.Hofstein i R.Heiman wytwarzają tranzystor polowy Mosfet	[USA]
1962	General Electric tworzy system operacyjny GECOS (później GCOS)	[USA]
1962	Powstaje "algebra informacyjna" - Algebraic Algebra, opracowana przez CODASYL-LSG w oparciu o prace R.Bossaka. W jej ramach zdefiniowano takie pojęcia jak "entity", "property", "value".	[USA]
1963	Publikacja książki Zdzisława Pawlaka "Organizacja maszyn bezadresowych", zawierającej wyniki jego pionierskiej pracy badawczej w latach 1952-1962	[Polska]
1962-1967	W tych latach Norwedzy Ole Johan Dahl i Kristen Nygaard opracowywują i implementują język SIMULA (SIMUlation LAnguage). Np. w 1965 r. na komputerze UNIVAC 1107 w USA. Początkowo traktowano go jako rozszerzenie języka ALGOL polegające na dodaniu równoległych procesów. W języku Simula zapoczątkowano podejście obiektowe w programowaniu (obiekty, klasy, dziedziczenie). Język ten występował potem najczęściej pod nazwą Simula 67.	[Norwegia]
1963	DEC (Digital Equipment Corporation) zapoczątkowuje modelem PDP-6 (Programmed Data Processor-6) produkcję minikomputerów. Tak naprawdę był to prototyp PDP-10.	[USA]
1963	W komputerze CDC 3600 zastosowano hardware'ową (mikroprogram?) implementację języka IPL-V, która była wykorzystywana do przyspieszenia komputerowej gry w szachy	[USA]
1963	W MIT zrealizowano projekt MAC (Multiple-Access Computer) i uruchomiono komputer pracujący w podziale czasu ("time-sharingu")	[USA]
1963	Ch.Bachman w GE opracowuje IDS (Integrated Data Store) - pierwsze oprogramowanie do zarządzania strukturami "łańcuchowymi" w plikach	[USA]

	(1sza wersja była rozszerzeniem języka symbolicznego, a potem COBOLu - czyli bez zastosowania schematu bazy danych)	
1964	Edsger W.Dijkstra rozwiązuje problem tzw. zakleszczenia występującego przy przydziale zasobów komputera (koncepcja semaforu)	[Holandia]
1964	IBM publikuje pierwszą wersję języka PL/1 (początkowo zwanego NPL - New Programming Language) przeznaczonego dla serii 360	[USA]
1964	W Dartmouth College John Kemeny i Thomas Kurtz opracowują język BASIC dla komputera GE225	[USA]
1964	W kwietniu 1964 r IBM ogłasza pojawienie się serii System/360, która zapoczątkowała nową klasę mainframe'ów (układy scalone, pamięć wirtualna od modelu 67)	[USA]
1965	IBM wypuszcza systemy operacyjne DOS/360 i OS/360	[USA]
1965	W PW powstaje UMC10 (tranzystorowa wersja UMC1 - w sumie wyprod. kilka egz.)	[Polska]*
1965	DEC wypuszcza minikomputer PDP-8.Do roku 1968 wyprodukowano ich 1450 egz.	[USA]
1966	Börje Langefors wydaje pracę "Theoretical analysis of information systems", zawierającą podstawy teorii systemów informacyjnych. Posiada wiele prac w dorobku, w tym "Information systems theory" (1977), "Infological models and information user view" (1980)	[Szwecja]
1966	Martin Richards z Univ.Cambridge opracowuje (głównie podczas pobytu okresowego w MIT) składnię języka BCPL (Basic Combined Programming Language), z którego wywodzi się linia języków C.	[UK]
1966	Istnieje już 1200 języków programowania, nie licząc mutacji ALGOlu, COBOLu, FORTANu itp. W 1963 r. występowało ok.300 języków, w tym 114 ukierunkowanych na obliczenia matematyczne,a 16 do zastosowań administracyjno-ekonomicznych (nie licząc 39 kompilatorów języka COBOL).	[Świat]*
1966	W IMM powstaje komputer ZAM41 (w sumie wyprod. 20 egz.)	[Polska]*
1967	O tym jak szybko rozwija się amerykański przemysł komputerowy świadczy fakt, że wysokość sprzedaży szacowana jest na ok.4 miliardy dolarów, czyli w ciągu 4 lat została potrojona (1963-1,3 mlrda)	[USA]*
1967	Jef Raskin (późniejszy twórca komputerów Macintosh w firmie Apple) pisze pracę doktorską n/t GUI (Graphical User Interface) na Uniwersytecie Pennsylwania	[USA]
1967	Powstają pierwsze pamięci - tzw.płaskie warstwy magnetyczne - zastosowane w komputerach UNIVAC 1107 i Burroughs 8500	[USA]
1967	W ELWRO powstaje komputer ODRA1204 (w sumie wyprod.180szt)	[Polska]*
1966	W 1966r Dijkstra kończy projekt wieloprogramowego systemu operacyjnego THE, a publikuje jego opis w 1968r.	[Holandia]
1968	Wprowadzenie na rynek statycznej pamięci RAM	[USA]
1968	Utworzono firmę Intel (Bob Noyce, Gordon Moore i William Shockley)	[USA]*
1968	W USA pracuje już ok. 68000 komputerów - świadczy to o dużym potencjale amerykańskiego przemysłu komputerowego i jego dochodach (komputer kosztuje kilkaset tysięcy dolarów.))	[USA]*

1968	Ogólna liczba komputerów pracujących w podziale czasu w USA wynosiła w 1964 r. tylko 10, a w r.1968 jest ich ok.100. Tyle więc przyrosło w ciągu 5 lat po uruchomieniu w 1963 r projektu MAC w MIT. General Electric (GE) zostaje liderem obsługi terminalowej (w aktualnej terminologii "on-line"), gdyż liczba takich klientów wynosi u niego ponad 50000. Jeszcze 3 lata przedtem liczba klientów pracujących na terminalach w całych Stanach wynosiła 500.)	[USA]*
1968	W liście do Edytora Communic.of ACM Dijkstra pisze o szkodliwości instrukcji GO TO. Zapoczatkowało to pracę nad dziełem "Structured Programming" autorstwa Dahl O., Dijkstra E.E. i Hoare C.O.R. wyd.w 1972r przez Academic Press. Tematykę tę kontynuował w USA m.i. E.N. Yourdon "Techniques of Program Structure and Design" (1975-Prentice Hall).	[Holandia]
1968	Adriaan van Wijngaarden opracowuje dwustopniową gramatykę (vW-grammar,W-grammar) do definiowania "infinite context-free grammars in a finite number of rules", wykorzystaną do definiowania jezyka ALGOL 68.	[Holandia]
1969	Po raz pierwszy w minikomputerze zastosowano wirtualną pamięć oraz zmienny przecinek - był to norweski 16-bitowy NORD-1 zbudowany w 1967 r.	[USA]
1969	Departament Obrony US uruchamia rozległą sieć ARPANET (Advanced Research Projects Agency Network)- poprzednik globlnego internetu (stworzył podstawy prac nad TCP/IP, rozpoczęte w 1973 r w ramach tego samego projektu DARPA)	[USA]
1969	W IBM s.370 zastosowano pierwszy floppy dysk - 8 calowy o pojemnosci 80KB, read-only, do wprowadzania mikrokodu	[USA]
1969	W MIT oddano do użytku wielodostępny system MULTICS	[USA]
1969	Kenneth Thompson i Dennis Ritchie z laboratoriów AT&T Bella tworzą podstawy systemu operacyjnego UNIX	[USA]
1969	Cincom Systems oferuje bazę danych TOTAL (typu CODASYL)	[USA]
1969	IBM wprowadza do sprzedaży bazę danych IMS	[USA]
1969-1973	Dennis Ritchie tworzy w Bell Teleph. Lab. język programowania C, równolegle pracując nad systemem UNIX. NAjbardziej owocny dla C był rok 1972. Dalszy rozwój C następuje w latach 1977- 1979	[USA]
1970-1971	Ted Hoff (wg swego pomysłu z 1968 r) w firmie Intel tworzy pierwszy 4-bitowy mikroprocesor krzemowy 4004, zawierający 2,3 tys. tranzystorów i taktowany sygnałem 100kHz	[USA]
1970	B. Metcalf i D. Boggs z ośrodka badawczego Palo Alto tworzą sieć Ethernet	[USA]
1970	Fairchild wprowadza na rynek pierwszą 256-bitową statyczną pamięć RAM	[USA]
1970	Intel opracowuje pierwszą pamięć dynamiczną RAM o pojemności 1024 bitów	[USA]
1970	Firma Corning Glass ogłasza wyprodukowanie światłowodu o tłumieniu poniżej 20 dB/km, co daje możliwość komercyjnego ich zastosowania	[USA]

1970	Powstaje system zarządzania bazą danych IDMS (Culliname)	[USA]
1970	E.F. Codd prezentuje model relacyjnej bazy danych	[USA]
1970-1972	Jay Wurtz i Rick Karrash, studenci Sloan Management School - potem w firmie IRI, budują Express - pierwszy produkt software'owy realizujący zapytania OLAP (online analytical processing) choć ten termin jeszcze wówczas nie istniał, na wielowymiarowej bazie danych. Wprowadzenie OLAPu zmodernizowało relacyjne bazy danych o schematy gwiazdy (star) i śnieżynki (snowflake), służące do tworzenia tzw. kostek wielowymiarowych.	[USA]
1970	Wprowadzenie przez IBM serii System/370	[USA]
1970	Lata 70-te w ELWRO wyprodukowano 50 egz. komputera Riad32	[Polska]*
1971	Ray Tomlinson wysyła pierwszy email przez ARPANET	[USA]
1971	Charles Moore opracowuje język Forth przeznaczony do sterowania radioteleskopem. W 1994 r uzyskał ten język standard ANSI. Zastosowano w nim odwrotną polską notację (Reverse Polish Notation, RPN)	[USA]
1971	Niklaus Wirth opracowuje język PASCAL.	[Szwajcaria]
1971	SAG (Software AG) wprowadza bazę danych ADABAS z listami inwersyjnymi	[Niemcy]
1971	Powstaje komputer ODRA1305 ELWRO (w sumie wyprod.500 szt ODRA1305/1325)	[Polska]*
1971	Hitachi opracowuje file-storage (dysk?) 1 GB [wg innych źródeł:dopiero w 1980 r IBM wprowadza na rynek pierwszy dysk gigabajtowy - waga ponad 200kg i cena ponad 40 tys.$]	[Japonia]
1971	Alan Shugart przenosi się z IBM do Memorex, gdzie opracowuje pierwszy read-write floppy dysk Memorex 650	[USA]
1971-1972	Na uniwersytecie w Marsylii Philippe Roussel i Alain Colmerauer tworzą PROLOG (franc. PROgrammation en LOGique) język programowania oparty na logice i lingwistyce komputerowej. Język ten został później rozwinięty przez logika Roberta Kowalskiego z Univ.w Edynburgu.	[Francja]

DODATEK 2

O TYCH CO STWORZYLI PODSTAWY INFORMATYKI I PRZEMYSŁU KOMPUTEROWEGO

To, co kryje się obecnie w pojęciu "informatyka" lub "computer science" nie urodziło się w jednym akcie jak dziecko. Narastało powoli i od dawna na kilku kontynentach (z "prehistorii" informatyki wymienić można takie postaci jak Pingala, Paniani, Leonardo

da Vinci, Blaise Pascal, Gottfried von Leibniz, Charles Babbage itd.] , ale niewątpliwie największy wkład w jej głównym dorobku jest dziełem zaledwie kilku lub najwyżej kilkanastu osób. Właśnie o nich - o wybitnych postaciach informatyki wszechczasów - będę opowiadać.
Do grona "wielkich" z wyżej wymienionych zaliczyć trzeba też Ch.Babbage'a, o którym pisalismy w rozdz.I. Przesłanki i pierwsze koncepcje automatycznego liczenia.

Herman Hollerith - symbol epoki maszyn liczących systemu kart dziurkowanych

Na poczatku tego dodatku warto wspomnieć Hermana Holleritha (1860-1929) –Amerykanina niemieckiego pochodzenia - twórcę, który zapoczątkował stuletnią epokę maszyn liczących systemu kart dziurkowanych, zwanych maszynami licząco-analitycznymi.
Występowanie Holleritha na wspólnej liście z Neumannem i Turingiem moze wydawać się niewłaściwe, ale zapewniam, że nie było to przeoczenie z mojej strony.
Po pierwsze, dał początek wielkiemu biznesowi w technice obliczeniowej gdy w roku 1896 utworzył "Tabulating Machine Company" (TMC), która łącząc się z kilkoma innymi firmami dała w roku 1924 początek największej firmy komputerowej w dziejach świata - IBM, zatrudniającejw latach 60-tych ponad 500 tysięcy osób.
Po drugie, był wybitnym wynalazcą - miał na koncie 38 patentów (w tym na elektryczne hamulce kolejowe), jednakże najważniejszy w jego mniemaniu patent pneumatycznego odczytu kart dziurkowanych nigdy nie został zrealizowany, gdyż nawet on w swoich konstrukcjach stosował przekaźnikowy odczyt elektryczny).
Po trzecie, - może najważniejsze - zastosował technologię *strumieniowego [a nie pojedynczego] masowego odczytu wielokolumnowych kart dziurkowanych* , dało znaczący impuls rozwojowy dla produkcji późniejszych komputerów.

Opatentował [U.S. Patent 395,782] wynaleziony przez siebie sposób elektrycznego odczytu karty, zaś samej karty jako nośnika danych nie mógł opatentować, gdyż na przeszkodzie stały takie historyczne fakty, jak używanie kart w przemyśle tkackim (pomysł z 1725r Basile'a Bouchona i Jean-Baptiste'y Falcona, następnie ulepszony przez Jaccarda) oraz idea Charlesa Babbage'a użycia kart dziurkowanych w maszynie analitycznej.
Natomiast tytuł doktora otrzymał w Columbia Univ. za skonstruowanie tabulatora [An Electric Tabulating System], czyli najważniejszej maszyny licząco-analitycznej, która sumowała dane z wielu kart. W latach 1915-1920 Powers [wprowadził 90-kolumnowe karty i rozpoczął własną produkcję maszyn] - wzbogacił tabulatory o mechanizm drukujący i danych sumarycznych nie trzeba było juz spisywac ręcznie, lecz można je było drukować poprzedzając danymi analitycznymi, które się na podsumowania składały.
Grupowanie danych [niezbędne do tzw. sumowania stopniowego: total, subtotal itp.] było mozliwe dzięki sorterom, które mogły również zliczać karty - notabene tego typu urządzenie było używane jeszcze niedawno w USA do liczenia głosów w wyborach prezydenckich [George W. Bush kontra Al Gore w 2000 roku].
Ale wracając do tabulatora, warto odnotować, że była to maszyna programowana przez tzw. komutatory kablowe (wiredboards -control panels -plugboards) i na nich wg wszelkiego prawdopobieństwa wzorowali się twórcy ENIACa. A więc już na pierwszym etapie budowy komputerów czerpano wzory z maszyn licząco-analitycznych.
Sama karta dziurkowana była z istoty swojej zapisem binarnym (0 - brak otworu, 1 -

otwór), co idealnie ją później kwalifikowało do użycia w komputerach i ułatwiło w latach 50-60 tych przenoszenie zastosowań z maszyn analitycznych na elektroniczne komputery (z możliwym użyciem tych samych kart do przetwarzania danych historycznych!).
To dzięki kartom stało się możliwe przetwarzanie danych ekonomiczno-administracyjnych, wymagające dla jednego przebiegu zwykle wielu transakcji (znajdujacych się na tysiącach kart czyli technologii świetnie opanowanej przy stosowaniu tabulatorów).
I tutaj dochodzimy do *największej zasługi historycznej Holleritha* - dał impuls i podstawy prawie 100-letniej epoki masowego prawie zunifikowanego przetwarzania danych w skali światowej w technice maszyn licząco-analitycznych, z której możliwe było płynne przejście do nowoczesnych technik obliczeniowych. Nie trzeba o tym zapominać, mimo iż maszyny te zniknęły już dawno nawet z wielu muzeów.
W latach 30-tych największy światowy potentat maszyn systemu kart dziurkowanych IBM zaczyna produkować kalkulatory (seria 600) znacznie rozszerzające zakres obliczeń (w stosunku do tabulatorów, co zapowiadało "rewolucję" w przetwarzaniu danych, ale wymagało uprzedniego postępu w elektronice. Przykładem bezpośredniego pionierskiego wykorzystania technologii kart w elektronicznej technice obliczeniowej są pochodzące z początku lat 50-tych komputery UNIVAC60 i UNIVAC120 firmy Remington Rand [notabene konkurenta TMC w produkcji maszyn licząco-analitycznych], będące w istocie swojej kalkulatorami systemu kart dziurkowanych wyposażonymi w pamięć elektroniczną na lampach próżniowych.

Mauchley czy Atanasoff czy Konrad Zuse

(czyli spór o to kto był pierwszy)

Podstawy konstrukcyjne komputerów stworzone zostały w czasie i po II wojnie światowej głównie na uczelniach amerykańskich (von Neumann w Princeton, Atanasoff w Iowa, Eckert i Mauchly w Pensylwanii, Howard Aiken w Harvardzie). Dorobek naukowy wchłaniany był chętnie przez przemysł, często wraz z ludźmi (np. przejście Eckerta i Mauchly'ego do Remington Rand dały początek działowi komputerów komercyjnych UNIVAC w tej firmie). Rozwój przemysłu komputerowego początkowo odbywał się w dużych firmach produkujących przedtem inne produkty (np. maszyny analityczne systemu kart dziurkowanych, maszyny do pisania, golarki elektryczne), a potem nabrał radykalnego przyśpieszenia dzięki inwencji kilku młodych prężnych i ambitnych osób (założycieli firm DEC, Apple, Dell). Od lat 70-tych postęp w konstrukcji komputerów zawdzięczamy już nie tyle uczelniom, ile takim wynalazcom-fascynatom jak Ken Olsen i Seymour Cray.

Presper Eckert i John Mauchly (twórcy ENIACa), John Atanasoff i Clifford Berry (twórcy ABC) oraz John von Neumann (twórca podstaw architektury komputerów)

spotykali się i wymieniali poglądy związane z budową pierwszych komputerów.
Neumann nie przywiązywał większej wagi do praw autorskich, traktując osiagnięcia naukowe jako wspólne dobro lub dzieło zbiorowe o licencji public domain, skoro projekt był finansowany przez rząd amerykański. Natomiast panowie Eckert i Mauchly jeszcze pracując nad ENIACiem weszli w spór ze swoim Uniwersytetem (Pensylvania Univ.) o wyłączność praw patentowych, związanych z konstrukcją tego komputera. Spór zakończył się przerwaniem z ich strony prac nad całkowicie binarnym komputerem EDVAC, odejściem z uniwersytetu i założeniem własnej firmy EMCC. Ponadto zarzucali

Neumannowi, że w swoich raportach wykorzystywał ich dorobek naukowy. W końcu zdobyli prawa patentowe, ale historia – za pośrednictwem Johna Atanasoffa (twórcy w 1942 roku komputera ABC –Atanasoff Berry Computer) rozliczyła właściwie kwestie ich pierwszeństwa. Ignorowany przez nich wynalazca wytoczył w 1971 roku proces o prawa autorskie i po 2 latach procesowania się uzyskał pełną satysfakcję – patenty związane z ENIACiem zostały uznane za nieważne a Atanasoff uznany został za wynalazcę pierwszego elektronicznego cyfrowego komputera. Niestety informacje o tym werdykcie nie zostały nalezycie rozpropagowane i pogląd o ENIACu jako pierwszym komputerze istnieje do tej pory nawet wśród osób jako tako zapoznanych z historią informatyki.

Po tych kontrowersjach wróćmy teraz do **J. von Nemanna.** Żył krótko (1903-1957), ale intensywnie, zajmując się – poza "computer science" - wieloma dziedzinami (m.i. teorią gier, fizyką nuklearną, teorią kwantów, a także wynalazkiem bomby wodorowej). Był Węgrem (urodził się w Budapeszcie), studiował chemię w Berlinie i Zurychu, doktorat z matematyki (teoria zbiorów) otrzymał na Uniwersytecie Budapeszteńskim. Wykładał w Berlinie i Hamburgu, a w wieku 27 lat wyemigrował do USA do Princeton, gdzie był jednym z założycieli Institute for Advanced Studies. Problemy z rozwiązywaniem nieliniowych równań różniczkowych w hydrodynamice skierowały go w stronę obliczeń komputerowych, najpierw do komputera Mark I (autorstwa Howarda Aikena), a potem ENIACa. Świadcząc usługi konsultanckie dla Eckerta i Mauchleya opracował takie fundamentalne zasady architektury komputerów jak:

- program ma być przechowywany w pamięci i poddawany przetwarzaniu (podobnie jak dane) np. w celu generowania następnych rozkazów

- 3 podstawowe części komputera to: pamięć (dla programu i danych), procesor (w ramach którego wydzielona bywa część sterująca oraz część arytmetyczno-logiczna) i urządzenia wejścia-wyjścia.

Neumann otrzymał 7 doktoratów honorowych i w 1956 roku nagrodę im.Enrico Fermiego.

Alan Turing - wybitny angielski matematyk i konstruktor Ukonczył angielski Cambridge, potem był zastępcą dyrektora laboratorium komputerowego w uniwersytecie w Manchesterze. Był wybitnym teoretykiem (Maszyna Turinga), badaczem sztucznej inteligencji ("test Turinga"), bohaterem II wojny światowej [był członkiem grupy naukowców z Blatchey Park, którzy pracowali nad łamaniem kodu niemieckiej maszyny szyfrującej Enigma] oraz ... gejem. Ta ostatnia cecha nie była tolerowana w epoce, w której żył i poniżony publicznie po przymusowej (wyrok sądowy) chemicznej kastracji prawdopodobnie popełnił samobójstwo w 1954 roku. W sierpniu 2009 roku grupa brytyjskich naukowców wystosowała petycję -podpisaną przez tysiace ludzi -do brytyjskiego rządu, w której domagała się przeprosin za śmierć Alana Turinga. Teoretyczna " Uniwersalna Maszyna Turinga" zdefiniowana już w 1936 roku świadczyła o tym, iż mimo iż potemuczestniczył w tworzeniu specjalizowanego komputera Colosssus, zdawał sobie sprawę, że komputery z istoty swojej są uniwersalne. Turing - podobnie jak Zuse - był raczej samotnikiem - pracującym w Anglii, nie nawiązującym mocnych więzi z nurtem amerykańskim, aczkolwiek poznał Neumanna w latach 1937-39 przebywając w Princeton i niezależnie od niego formułował podobne poglądy (np. w pamięci komputera powinny znajdowac się zarówno dane jak i program) co tzw. Raport Grupy Neumanna. Stworzony przez niego system kodowania (base 32 coding) został wykorzystany do zapisu na taśmie papierowej zarówno programow jak i danych w komputerze MARK 1 [nie mylic z przekaźnikowym komputerem Aikena o tej samej nazwie w amerykańskim Harvard Univ.], którego inna wersja potem nosiła nazwę

Ferranti MARK 1. Komputer ten -realizowany najpierw jako Small-Scale Experimental Machine (SSEM)- był prawdopodobnie pierwszym (1948-1949)elektronicznym komputerem przechowującym program w pamięci [była ona typu CRT] i wyposażonym w pamięć na bębnie magnetycznym. Ferranti Mark 1 uważany jest przez niektórych za 1szy uniwersalny komputer sprzedawany komercyjnie.

W gronie wynalazców komputera nie można pominąć Howarda Aikena i jego współpracownicy Grace Hopper.

Opracowywali oni komputery MARK na uniwersytecie Harvard. Zbudowany na przekaźnikach i uruchomiony w 1944 r. MARK I, znany również pod nazwą ASCC -Automatic Sequence Controlled Computer, był pierwszym automatem liczącym w USA sterowanym przez program. Wykorzystywano go do obliczeń balistycznych w US Navy do roku 1959 !. Uruchomiony w 1951 roku MARK III oprócz 2000 przekaźników, miał już 5000 lamp elektronowych i 1300 diod.

Howard Aiken (1900-1973) był z wykształcenia inżynierem elektrykiem i fizykiem. W 1937 roku rozpoczął pracę nad komputerem MARK I, a w 1939 roku uzyskał tytuł doktora. Prace badawcze nad komputerem zaczęła finansować firma IBM i Aiken stanął na czele 3-osobowego zespołu, w skład którego wchodziła G.Hopper. W 1947 rok Aiken utworzył Harvard Computation Laboratory, a potem własną firmę Aiken Industries. Był zakochany w komputerach, ale nie przeczuwał ich późniejszego szerokiego rozprzestrzenienia. Uważa się, że do niego należy powiedzenie z 1947 roku: "Tylko sześć elektronicznych cyfrowych komputerów wystarczy do zaspokojenia potrzeb obliczeniowych całych Stanów Zjednoczonych." Notabene podobne proroctwo z 1943 roku dotyczy Thomasa Watsona (wówczas prezesa IBM) - " Myślę, że dla całego świata wystarczy być może 5 komputerów".

Jedyną do tej pory kobietą w gronie znaczących twórców informatyki była **Grace Brewster Murray Hopper** (1906-1992), oczywiście nie licząc legendarnej córki poety Ady Byron - Lady Lovelace - wielbicielki maszyny Babbage'a, która opracowując dla niego procedurę obliczania liczb Bernouliego uchodzi za pierwszą programistkę. G.Hopper w cywilu była profesorem matematyki, a podczas wojny w 1943 roku na własne życzenie została pracownikiem marynarki (w Naval Reserve). Współpracę z Aikenem rozpoczęła w 1944 roku w końcowej fazie budowy komputera Mark I , niewątpliwie w związku z tym, że głównym użytkownikiem tego komputera miała być marynarka. Hopper była odpowiedzialna za testowanie komputera i wyłapywanie błędów . To ona właśnie jest autorem terminu "debug" (odpluskwiania).

W 1949 roku rozpoczęła pracę nad kompilatorem języka symbolicznego dla firmy EMCC (Eckert-Mauchly Computer Corporation), a następnie w Remington Rand/Sperry Rand była wiodącym członkiem zespołu, który dla komputera Univac opracował Flow-Matic jeden z pierwszych języków programowania rozpoznający komendy słowne (nie symboliczne) w języku angielskim. Następnie opracowała język APT i była odpowiedzialna za weryfikację języka COBOL, którego współautorem była Betty Holberton. Uhonorowano ją w 1969 roku po raz pierwszy wprowadzoną nagrodą "Postać Roku - Man of the Year" w dziedzinie computer science. Pod koniec życia w 1991 roku otrzymała medal "National Medal of Technology". Służbowy kontakt z marynarką utrzymywała prawie do końca życia – mimo iż była od dawna w wojskowym wieku emerytalnym (najpierw zgodnie z prawem przymusowo skierowano ją na emeryturę, a potem przyjmowano z powrotem gdyż była niezastąpiona). Między innymi uznano ją za niezbędną do wdrażania języka COBOL i standaryzacji oprogramowania w marynarce. W 1983 roku Prezydent USA mianował ją komandorem, a dwa lata później w wieku 79

lat (!) została pierwszą kobietą admirałem. W 1986 roku po 43 latach służby w marynarce przeszła wreszcie uroczyście na emeryturę. Nie zerwała jednak z informatyką, gdyż przez ostatnie lata życia była konsultantem w Digital Equipment Corporation. Jej życie było piękną wzorcowo zapisaną kartą, jakiej można życzyć każdemu informatykowi.

KONRAD ZUSE (1910-1995)

Konrad Zuse był bez wątpienia pionierskim wynalazcą komputerów. Działał prawie samotnie, jakby na uboczu głównego nurtu amerykańskiego i angielskiego, obejmującego takie postacie jak John von Neumann, Alan Turing i Howard Alken. Jego rozwiązania funkcjonalne były zbieżne (z punktu widzenia architektury komputera, za wyjątkiem takich szczegółow jak obsługa instrukcji warunkowych oraz przechowywanie programu w pamięci - miała ona w Z3 tylko 64 słowa) z raportem Grupy Neumanna. Komputer Z3 ten miał modułową budowę: jednostkę sterującą, pamięć, jednostkę arytmetyczną, urządzenia wejścia-wyjścia. Niestety nie był elektroniczny, lecz zbudowany na przekaźnikach - i to zaważyło na pomijaniu go na liscie pierwszych komputerów - ale nie powinno decydować o pomijaniu rangi jego koncepcji architektonicznych (nad ktorymi pracował od 1934 roku) i pierwszeństwa w opracowaniu języka programowania (w latach 1942-1945 opracował język Plankalkül).
Dyskusji nie podlega, że Z3 był pierwszym programowalnym komputerem, przechowującym program na taśmie papierowej.

Pierwszym elektronicznym komputerem Zusego był Z22, zbudowany w 1955 roku na lampach próżniowych. Było to zbyt późno, aby zasługiwało na szczególne uznanie, ale świadczyc może o trudnosci w dostępie do nowoczesnych technologii w powojennych Niemczech.

Zapewne do izolacji przyczyniły się ograniczenia informacyjne, wynikające z prowadzenia II wojny światowej (zarówno ENIAC jak i Z3 miały służyć prawdopodobnie celom wojennym).Sam Konrad Zuse pracował nad zdalnie kierowanym pociskiem rakietowym Henschel Hs 293, który w 1943 roku zatopił amerykański okręt HMT Rohna z ponad 1000 osobami na pokładzie. Założona przez niego firma Zuse-Apparatebau pracowała nad specjalnymi komputerami do projektowania samolotów i profili aerodynamicznych.

W kręgach światowych i źródłach amerykańskich Zuse jako wynalazca był długo nieobecny, dopiero konferencja zorganizowana już po śmierci wynalazcy przez jego syna Horsta -na znak protestu przeciwko ignorowaniu zasług ojca - miała nagłośnić je w skali międzynarodowej.

Konrad Zuse był równocześnie wynalazcą biznesmanem: założone przez niego firmy wyprodukowały kilkaset komputerów Z11, Z22, Z23 i Z25.

ZAŁOŻYCIELE FIRM KOMPUTEROWYCH - WYNALAZCY I BIZNESMENI

Kenneth H. Olsen – wynalazca minikomputerów i kreator dużej firmy

Ken Olsen już od wczesnej młodości miał pociąg do elektroniki. Jako chłopiec naprawiał w piwnicy radioodbiorniki, po wojnie studiował w MIT (1952 – M.S.), a następnie pracował na tej uczelni w Lincoln Laboratory, gdzie skonstruował kilka układów do wczesnotranzystorowych mainframe'ów TX-0 i TX-2. W latach 60-tych uzyskał kilka patentów wynalazczych (m.in. na pamięć ferrytową, bufor drukarki wierszowej, układ diodowej bramki transformatorowej itp.).

W 1957 roku za 70 tys. dolarów otrzymanych od generała Doriot z American Research and Development Corporation założył wraz z Harlanem Andersonem (z którym razem opuścili Lincoln Laboratory) firmę Digital Equipment Corporation (DEC), która rozpoczęła działalność od wytwarzania układów logicznych do testowania urządzeń elektronicznych. Olsen szybko doszedł do wniosku, że wiele zadań realizowanych przez mainfraimy można wykonać na mniejszych komputerach i jego firma z własnych układów skonstruowała w 1960 roku pierwszy minikomputer PDP-1 (Programmed Data Processor). W ten sposób wprowadził firmę na drogę pełną sukcesów (stała się drugą na świecie po IBMie firmą komputerową wycenianą na 14 mld dolarów) i zbudował podstawę swojej 35-letniej kariery przedsiębiorcy. Minikomputery kosztowały wielokrotnie mniej niż mainframe'y i zdobyły (m.i. dzięki promocjom ze strony DEC) mocną pozycję na uczelniach amerykańskich. Pozycja DECa wydawała się być niezagrożona i zbyt nieograniczony. Na przykład minikomputer PDP-11 (pierwszy 16-bitowy komputer) produkowany był od roku 1970 nieprzerwanie przez 20 lat mimo trwającego postępu technicznego. Następcą jego były 32-bitowe VAXy, a w latach 90-tych 64-bitowe ALPHy. DEC zawsze znajdował się w czołówce technologicznej.

W 1986 roku magazyn Fortune uznał go za najlepszego przedsiębiorcę w historii amerykańskiego biznesu. Olsen stosował w swojej korporacji nowoczesne metody zarządzania (m.i. wprowadził macierzowe struktury zarządzania, polegające głównie na podziale firmy na współpracujące ze sobą niezależne działy), lecz nie przewidział tego, że przyszłość przemysłu komputerowego tkwi w stacjach roboczych i komputerach osobistych. To on był autorem niefortunnej wypowiedzi w 1977 roku, że nie widzi przyczyny, dla której ludzie mieli być mieć komputery w domach (później wyjaśniał, że chodziło mu o to aby komputer nie ingerował zbytnio w życie domowe).

Kłopoty w DEC zaczęły się pod koniec lat 80-tych, kiedy Sun Microsystems zaczął zdobywać rynek swoimi stacjami roboczymi opartymi na Unixie. Powstała w 1982 firma umiała trafić w trendy rozwojowe (architektura RISC, symetryczne wieloprzetwarzanie, technologia Javy) i znalazła dla siebie miejsce – swoją niszę rynkową – produkując komputery działające pod UNIXem (z własnym systemem operacyjnym Solaris) w środowisku sieciowym (stacje Sparc). SUN umiał również nawiązać alianse partnerskie z czołowymi producentami oprogramowania bazodanowego (Informix, Ingres, Oracle, Sybase) poszerzając w ten sposób potencjał zastosowań swoich komputerów. SUN produkował sprzęt oraz nie zrezygnował z prac rozwojowych. Można powiedzieć, że dzięki otwartości na to co dzieje się w świecie informatyki, SUNowi udało się to, na czym potknął się Digital. Można więc wyrazić uznanie dla czwórki założycieli tej firmy (Scott McNealy, Vinod Khosla, Bill Joy, Andy Bechtolsheim).

Od 1992 roku firma Digital zaczęła ponosić poważne straty np. w 1994 roku 2 mld dolarów. Prawdopodobnie nakłady na prace rozwojowe były za duże (np. na unixowy system Ultrix) w stosunku do starań o zdobycie rynku. Zarząd widział konieczność gruntownych zmian i poprosił Olsena o rezygnację. Tak oto zakończyła się jego kariera przedsiębiorcy i wynalazcy, założyciela i CEO korporacji. Po odejściu Olsen nie wziął rozbratu z informatyką i poprowadził małą 40-osobowę firmę informatyczną, zajmującą się konsultingiem i dostawą sprzetu komputerowego.

Błąd strategiczny Olsena zaważył fatalnie na firmie, mimo iż od początku kładziono w niej nacisk na prace rozwojowe, zarówno w konstrukcji komputerów jak i oprogramowaniu, przyczyniając się w znacznym stopniu do rozwoju systemów operacyjnych (w tym do Unixa - poza własnym systemem OpenVMS) i takich urządzeń jak router, klastry komputerów oraz protokół ethernetu. W 1995 roku zanosiło się na

to, że firma wychodzi z dołka dzięki rozpoczęciu produkcji procesorów ALPHA, gdyż zdołała wygospodarować zysk 122 ml $. Ale kiedy w Digitalu zaczęto produkować komputery osobiste, wystąpił ogólnoświatowa tendencja spadku opłacalności produkcji tych komputerów. Nie pomogła wyprzedaż różnych działów, w tym sprzedanie Intelowi sztandarowych procesorów Alpha. W 1998 roku Digital został przejęty przez Compaqa – firmę powstałą dopiero w 1982 roku i produkującą to, czego nie chciał kiedyś Olsen, czyli komputery osobiste. Rozwojem przemysłu komputerowego zaczęły rządzić jednak twarde prawa rynkowo-kapitałowe. Przekonał się o tym sam Compaq wchłonięty niedługo potem (w 2001r) przez Hewlett-Packard.

Dwóch Stevów (Jobs,Woz)

Zaczynali w garażu i konsekwentnie trzymali się własnej koncepcji rozwojowej

W 1976 roku dwóch Steve'ów – Wozniak (Woz, Wizard of Woz) i Jobs – oraz Ronald Wayne (pracował tylko przez krotki okres w Apple) w garażu wyprodukowali swój pierwszy komputer Apple I. Ponieważ zatrudniająca Woza firma Hewlett Packard nie zainteresowała się maszyną, na której można było uruchamiać BASIC, założyli 3 stycznia 1977 roku firmę Apple. W ciągu roku sprzedali 200 egzemplarzy i rozpoczęli wytwarzanie Apple II. Wtedy na rynku mikrokomputerów królowała firma Commodore (4 milionów użytkowników w 1984 r.). Woz zademonstrował jej prototyp Apple II , lecz firma nie wykazała żadnego zainteresowania (!). Na sukces początkowy firmy Apple zapracowali obaj – inżynieryjne umiejętności Wożniaka i marketingowe zdolności Jobsa (który już miał pewne doświadczenie komputerowe z firmy Atari). Nie można pominąć Mike'a Markkula, który na początku zainwestował w firmę prawie 100 tysięcy dolarów.

W 1985 Jobs odszedł z Apple, zakładając nową firmę – NeXT, która została przejęta przez Apple w 1996 roku, sprowadzając Jobsa z powrotem do macierzystej firmy, która zresztą zaczęła podupadać finansowo. Jobs odmienił sytuację dzięki wprowadzeniu na rynek komputera iMac, a następnie odtwarzacza iPod i internetowego sklepu muzycznego iTunes Store.

Los nie oszczędzał Woza. Jego zmysł wynalazcy nie kończył się nigdy.W 1981 roku rozbił się pilotowany przez niego samolot. Po hospitalizacji wyciągnął z tego praktyczne wnioski i zabrał się samodzielnie do projektowania automatycznego pilota. W Apple ustąpił pole Jobsowi, stając się pracownikiem z niewielkim wynagrodzeniem. Był raczej indywidualistą (chciał robić przede wszystkim to, co chciał sam) i praca w dużym zespole hamowała jego twórcze pomysły. Coraz mniej pracując w Apple, coraz bardziej angażował się w działalność charytatywną - założył w San Jose fundację Electronic Frontier Foundation, ufundował także muzeum Tech Museum of Innovation, wspierał finansowo budowę sieci komputerowej dla szkół. W roku 2001 założył firmę Wheel of Zeus, której zadaniem była popularyzacja platformy satelitarnego pozycjonowania GPS.

Apple miał znaczne zasługi w tworzeniu podstaw komputerów osobistych – to ta firma wprowadziła w 1984 roku po raz pierwszy GUI (graphical user interface), uwalniając użytkowników od ubogiej komunikacji znakowej znanej z DOSa. Osiągnięcie to było w 1988 roku przedmiotem sporu z Microsoftem a proces trwał aż do 1997 roku i zakończył się niczym (żadna ze stron nie była w stanie udowodnić racji).

Seymour CRAY

(1925-1996)

Wynalazca ciągle poszukujący lepszych rozwiązań technicznych i biznesowych – przykład "autorskiej" drogi innowacyjnej w przemyśle komputerowym

Karierę zaczynał w 1950 roku jako inżynier w Engineering Research Associates (ERA), gdzie pracował nad komputerem ERA 1101 przeznaczonym dla rządu amerykańskiego a potem nad innymi modelami. W 1957 roku wraz z kilkoma kolegami z ERY założył firmę CDC (Control Data Corporation) i tam był wiodącym konstruktorem kilku wówczas największych pod względem mocy obliczeniowej komputerów ogólnego przeznaczenia (np. chłodzonego freonem CDC 6600). Przedmiotem jego zainteresowania były nie tylko rozwiązania hardware'owe - opracował np. zestaw instrukcji programowych, które później określano jako RISC. Był uważany za "geniusza do szybkich komputerów". Wyprzedzał możliwości przemysłu komputerowego - to on skonstruował pierwszy superkomputer wielowątkowy i wieloprocesorowy CDC8600, którego nie udało się dokończyć z powodu zbyt dużych nakładów. Do jego osiągnięć należy Cray T3E-1200E, który jako pierwszy osiągnął moc obliczeniową 1 teraflop/sek.

Być może dlatego założył w 1972 roku firmę Cray Research, Inc. (w latach 1989-1995 Cray Computer Corp.), w której w 1976 roku wyprodukował superkomputer Cray-1. Piastował tam prawie wszystkie możliwe stanowiska kierownicze (dyrektora wykonawczego CEO i prezesa). W 1981 roku zrezygnował ze wszystkich stanowisk i poświęcił się pracy nad superkomputerem Cray-2 (o 10ciokrotnie większej szybkości niż jego poprzednik) i nowymi technologiami (gallium arsenic technology). Skonstruował jeszcze Craya-3, jednakże z powodu trudności technologicznych, opóźnień i braku sukcesów komercyjnych opuścił firmę.

Wpływ to miała zapewne chwila, kiedy uświadomił sobie potencjał mikroprocesorów i ich niski koszt w stosunku do procesorów projektowanych indywidualnie. O wartości mikroprocesorów przekonał się już wcześniej, kiedy używał komputerów Apple do symulowania superkomputera Cray-3. W celu wykorzystania tej technologii założył następną firmę autorską SRC Computers, Niestety nie zdążył zrealizować następnego pomysłu, gdyż zmarł w wieku 71 lat w wyniku obrażeń odniesionych w wypadku samochodowym w 1996 r w Colorado Springs.

Edgar Frank Codd

(1923-2003)

Twórca teorii relacyjnych baz danych z niespełnionymi nadziejami

Tak jak S.Cray był pionierem w konstruowaniu superkomputerów, to zasługą E.F.Codda było stworzenie teoretycznych podstaw relacyjnych baz danych, które stały się siłą napędową obecnego dużego biznesu (kilkunastu miliardowego-licząc w dolarach), skoncentrowanego na danych. Ale nie przyszło mu łatwo. Jego teorie na przełomie lat 60/70 tych za bardzo wyprzedzały ówczesną technologię, kiedy królowały sieciowe (np. typu DBTG Codasyl) i hierarchiczne bazy danych (np. IMS firmy IBM). A przecież nie proponował rzeczy bardzo złożonych, lecz raczej proste tablice składające się z wierszy i kolumn, które obudował pewną notacją oraz regułami tzw.normalizacji przeciwdziałającej dublowaniu danych. . Zasługą jego jest to, że uznał iż w obsłudze zapytań lepiej stosować relacje oparte na wartości danych, niż popularne wówczas "łańcuchowanie" danych poprzez pointery wbudowane w rekordy. Po prostu nie uwierzono w tak proste rozwiązanie. Stworzenie pojemnych pamięci operacyjnych i dyskowych umożliwiło efektywną realizację tej technologii.

Studiował matematykę i chemię na uniwersytecie w Oksfordzie. W czasie II wojny

światowej służył jako pilot w siłach powietrznych Wielkiej Brytanii. Po wojnie przeniósł się do USA. Pracował na początku jako programista w IBM. Na uniwersytecie stanu Michigan w Ann Arbor jako stypendysta IBMu uzyskał tytuł doktora "computer-science", a następnie zamieszkał , w Kalifornii, gdzie ponownie podjął pracę dla IBM.

W 1970 wydał pracę "A Relational Model of Data for Large Shared Data Banks", w której przedstawił relacyjny model zarządzania bazami danych. Pracodawca jego – IBM – nie przyjął z entuzjazmem jego pracy i nie kwapił się z jej praktycznym wykorzystaniem, kierując się prawdopodobnie interesem dotychczasowej sztandarowej bazy danych IMS. IBM nawet odsunął Codda od prac nad Systemem R i nie wykorzystał jego relacyjnego języka Alpha, lecz opracowywał język Sequel. Codd uważał ten język za niezgodny ze swoją teorią, co w efekcie zaowocowało odejściem z IBM i założeniem (wspólnie z Chrisem Date) firmy consultingowej. W latach 90-tych z powodu pogorszenia stanu zdrowia przestaje pracować.

Teoria relacyjnych baz danych czekała prawie 10 lat na komercyjną realizację. W 1979 roku Larry Ellison – założyciel firmy Relational Software (przemianowanej 6 lat później na ORACLE) na bazie teorii Codda –stworzył Larry Ellison (wraz z Bobem Minerem i Edem Oatesem) pierwszą komercyjną relacyjną bazę ORACLE. IBM pracował nad eksperymentalnym Systemem R i językiem Sequel, a dopiero w 1983 roku ogłosił opracowanie DB/2. Jednym z głównych ośrodków badawczych nad relacyjnymi bazami danych był Uniwersytet Berkeley (Ingres i język Quel) i to jego naukowcy utworzyli firmę Relational Technology, promującą Ingres. Quel się nie przyjął (choć był bardziej strukturalny), a Sequel (Structure English Query Language) z praktycznych względów został przemianowany na prostszą nazwę SQL.

Codd ma też poważne zasługi w innej dziedzinie bazodanowej, dotyczącej hurtowni danych, a więc technologii wielowymiarowych danych To on w 1993 roku ukuł termin OLAP (online analytical processing) i sformułował dwanaście reguł tej technologii.

Jednakże za twórcę koncepcji hurtowni danych uważany jest Bill Inmon, w latach 90-tych twórca firm Prism Solutions i Pine Cone Systems tworzących software do administrowania danymi w środowisku hurtowni danych. Sformułował on takie cechy hurtowni danych jak zorientowanie na podmioty (np. produkty, klienci) uwzględnienie wymiaru czasu, nieulotność i integralność danych, ukierunkowanie użytkownika końcowego na wspomaganie decyzji nie zaś na obsługę transakcyjną. Przy okazji warto dodać, że podstawy teorii wielowymiarowych baz zapoczątkowali w 1972 r Jay Wurtz i Rick Karrash, wówczas studenci Sloan Management School, a potem twórcy pakietu Express, pierwszego produktu pracującego tak naprawdę na technologii olapowej (choć ten termin jeszcze wówczas nie istniał). Wprowadzenie OLAPu zmodernizowało relacyjne bazy danych o schematy gwiazdy (star) i śnieżynki (snowflake), służące do tworzenia tzw. kostek wielowymiarowych.

Linus Benedict Torvalds (ur. 1969)

Bezinteresowny twórca - wystarczy chcieć i móc – czyli o narodzinach linuxa.

Już od wczesnych lat miał "skłonności" do komputerów. Najpierw był to dziadkowy Commodore Vic 20, a w wieku 18 lat kupuje za oszczędności 32-bitowy Sinclair QL z procesorem Motorola 68008, 7MHz, 128KB pamięci. Chęć doskonalenia systemu operacyjnego w Sinclairze nie mogła zostać zrealizowana, bo ten był wbudowany w ROM. W 1988 roku wstępuje na Uniwersytet Helsiński i tutaj w 1990 roku uczy się języka C, w którym wkrótce rozpoczyna pisać swój system operacyjny wzorowany na

Unixie i MINIXie (mały klon UNIXA opracowany przez Andrew Tanenbauma w Holandii do nauki studentów). Do DOSu Linus zraził się wystarczająco po zakupie komputera osobistego typu IBM.

Tworzy w swoim wolnym czasie przy użyciu własnego sprzętu i nawet robi sobie przerwę w studiach aby skoncentrować się na zadaniu. Już w 1991 roku ukazuje się jądro systemu. Torvalds początkowo nazwał system Freax (Free uniX) ale za sugestią przyjaciela Ari Lemmke nazwa została zmieniona na Linux (LINUs uniX).

Linux szybko zyskiwał popularność. W 1997 roku używany był już na ponad 3 milionach komputerów, a dwa lata później było ich już 7 ml. W tymże roku Linus Torvals – po 10 latach pobytu na Uniwersytecie w Helsinkach jako student i wykładowca – zdecydował się na wyjazd wraz z rodziną do doliny krzemowej. W Santa Clara w firmie Trameta pracował za przeciętną płacę programisty przy interfejsie pomiędzy systemami operacyjnymi (ale nie Linuxem) i mikroprocesorami tej firmy. Porozumienie z pracodawcą gwarantowało mu możliwość zajmowania się też Linuxem w celach nie związanych z firmą. Pewnym impulsem do wyjazdu była konieczność zarobkowania dla utrzymywania rodziny.

Praca Torvaldsa uzyskała przychylność nie tylko ze strony szerokiej szerzy współpracujących wolontariuszy programistów, lecz też niektórych ludzi biznesu, jak np. Paula Allena współzałożyciela Microsoftu, który udzielił mu wsparcia finansowego. Sam napisał decydujące 2% kodu jądra, co jest zrozumiałe, gdyż całość jest ogromna – wynosi ok.80MB, nie licząc wielu programów (takich jak GCC, edytor vi, XWindow System i KDE) opracowywanych poza ścisłym projektem Linuxa, lecz koordynowanych wspólnie.

Jego sytuacja finansowa radykalnie polepszyła się w 1999 roku – został milionerem - kiedy firmy Red Hat i VA Linux (VA Software) , budujący linię Linuxa dla dużych przedsiębiorstw w dowód zasług podarowały mu akcje o ówczesnej wartości 20 ml dolarów.

Obecnie Torvalds pracuje na pełnym etacie w OSDL (Open Source Development Lab), założonym w 2000r i finansowanym przez konsorcjum firm komputerowych (m.in. IBM rocznie wykłada ok. 2 miliardy dolarów) oraz takie firmy software'owe jak Oracle, Intel, Netscape i Corel. Jest to oznaką, że Linux został doceniony jako system innowacyjny i należy go rozwinąć do obsługi dużych firm, co przekłada się m.in. na możliwości zarządzania bardzo obciążonymi serwerami (Linux może pochwalić się sukcesem w postaci serwerów webowych Apache) . W ramach OSDL Torwalds podejmuje decyzje końcowe odnośnie zgłaszanych modyfikacji i uzupełnień, delegując pewne uprawnienia takim współpracownikom jak Alan Cox, Andrew Morton i Marcelo Tosatti Do niego należy marka Linux, aczkolwiek samo oprogramowanie jest licencjonowane jako otwarte (GPL), co nie zabrania sprzedawania go dla zysku.

Linus T. wyznawał zasadę wolnego kodu źródłowego, nie obwarowanego patentami i licencjami oraz dostępnego do modyfikacji. Już wczesną wersję udostępnił na serwerze FTP w katalogu Linux). Na zarzut przedstawiciela Microsoftu Craiga Mundiego, że "open-source" niszczy własność intelektualną , odpowiedział następująco:

"Zastanawiam się, czy Mundie słyszał kiedykolwiek o Izaaku Newtonie. Jest on sławny nie tylko ze względu na utworzenie podstaw mechaniki (i początki teorii grawitacji, o czym pamięta większość ludzi, łącznie z historią o jabłku), ale również z powodu stwierdzenia: ` Jeśli byłem w stanie spojrzeć dalej, to tylko dlatego, że stałem na ramionach olbrzymów'.

Jestem skłonny słuchać raczej Newtona niż Mundiego. Choć umarł prawie trzysta lat temu, mniej zasmradza pokój."

Linus Torvalds jest jeszcze prawie młodym człowiekiem, ale uhonorowany został już wieloma wyróżnieniami:

- w 1996 roku asteroida 9793 została nazwana jego imieniem
- w 1998 roku otrzymuje nagrodę EFF Pioneer
- w 1999 roku uzyskuje doktorat honorowy Uniwersytetu w Sztokholmie (warto dodać, że pochodzi ze szwedzkojęzycznej rodziny fińskiej), a w roku następnym Uniwersytetu w Helsinkach
- w 2004 roku uplasował się na 16 miejscu na liście 100 najsłynniejszych Finów wszechczasów
- w 2006 roku Time Magazine nazwał go jednym z najbardziej znaczących (revolutionary) bohaterów 60-lecia.

Wiliam (Bill) H.Gates (ur. w 1955 r)

Największy biznesmen i filantrop – człowiek któremu się powiodło

Można powiedzieć, że o ile Linus Torvalds pisał Linux dla przyjemności (powiedział: "I did it all just for fun"), to Bill Gates działał przede wszystkim dla biznesu.

Przez kilkanaście lat był najbogatszym człowiekiem w świecie. Wartość jego majątku wahała się w zależności od notowań giełdowych Microsoftu (posiada kilkanaście procent akcji) od 43 do ponad 59 miliardów dolarów.. Ostatnio na 1szym miejscu został zastąpiony przez meksykańskiego biznesmena Carlosa Slima. (ponad 65 mld), , posiadacza 1/3 udziałów w spółce telekomunikacyjnej America Movil (to czołowy operator sieci komórkowej w Ameryce Łacińskiej)..Był założycielem Microsoftu, od początku jej szefem, a potem prezesem Rady Nadzorczej i Głównym Architektem Oprogramowania. Jego firma ma 50 miliardów rocznego dochodu i zatrudnienia 78 tysięcy osób w 105 krajach.

W 1973 roku rozpoczął studia w Harvardzie, które przerwał 2 lata później, rozpoczynając partnerski biznes Micro-Soft z kolegą ze studiów Paulem Allenem. Już wcześniej zresztą obaj założyli firmę Traf-O-Data, do obliczeń dla potrzeb zarządzania ruchem drogowym, wykorzystując interpreter Basica, który opracowywali jeszcze podczas studiów na komputer Altair dla firmy MITS. Interpreter Basica był pierwszym masowym produktem dostarczanym w ciągu następnych paru lat dla prawie wszystkich mikrokomputerów amerykańskich i japońskich. Los sprzyjał Gatesowi. Niedługo potem IBM ogłosił planowaną premierę PC i pilnie potrzebował dla niego systemu operacyjnego. Microsoft podjął się realizacji zadania, wykupując od firmy Seattle Computer Products prawa do QDOSa (Quick'n'Dirty Operating System) i zatrudnił jego autora Tima Pattersona. QDOS został udoskonalony i jako **MS-DOS** przekazany firmie IBM do testowania. W lutym 1981 system został uruchomiony na prototypie IBM PC i w listopadzie ostatecznie zaakceptowany jako podstawowy system operacyjny dla nowych komputerów .

W następnych latach czołową rolę w biznesie komputerowym zaczęły odgrywać serwery. I tutaj Microsoft znalazł rozwiązanie. W 1988 r. zatrudnił Dave'a Cutlera, który opracowywał systemy operacyjne VAX/VMS i RSX-11M dla Digitala, a w Cutler zajął się 32-bitowym Windows NT. Innym produktem napędowym stał się pakiet biurowy MS

Office i następne wersje systemu Windows. Firma jest też producentem encyklopedii Encarta, symulatora lotu MS Flight Simulator, a nawet klawiatur czy myszek komputerowych oraz konsoli do gier (Xbox). Tyle w skrócie o działalności Microsoftu i tylko tyle, ile potrzeba było aby zilustrowac mechanizmy napędowe firmy.

A teraz o drugiej stronie medalu, czyli działalności charytatywnej państwa Melindy i Billa Gatesów. Od czerwca 2008 roku Bill chce poświęcić się całkowicie pracy w Bill & Melinda Gates Foundation nad poprawą stanu zdrowia i edukacji w świecie. Fundacja ta została przez małżeństwo Gatesów założona w styczniu 2005 roku z wkładem prawie 29 miliardów dolarów, a więc wielokrotnie większym niż zaangażowanie dotychczasowego największego amerykańskiego filantropa Johna D. Rockefellera .

Wreszcie wspomnimy o słynnych wpadkach Billa*: "640 kilobajtów pamięci powinno wystarczyć każdemu*" (wypowiedziane w 1981 roku) oraz później o Javie *"Ktokolwiek myśli, że mały, składający się z 9000 linii program, który na dodatek jest rozprowadzany za darmo i może być sklonowany przez każdego, zmieni cokolwiek, co robimy w Microsofcie, musi mieć pomieszane w głowie"*. Wizjonerem dobrym więc nie był, ale wszystko za co się brał w życiu kończył sukcesem.

Podsumowanie opowieści o gigantach informatyki może być krótkie:

- Herman Hollerith - symbol nadchodzacej epoki nowoczesnych maszyn liczących
- Konrad Zuse - wynalazca "izolowany" na obszarze Niemiec
- John von Neumann - światowy uczony
- Alan Turing - wybitny matematyk i teoretyk informatyki
- Mauchly - naukowiec i biznesman
- Seymour Cray - nieustanny wynalazca (aż do smierci)
- Linus Torvalds - bezinteresowny wynalazca
- Wiliam Gates - światowy biznesman

Jak trudno być prorokiem we własnym domu (czytaj: własnym biznesie) świadczą przepowiednie w latach 60-tych ub. stulecia: 5-6 komputerów zaspokoi świat, a 640 KB pamięci jest wystarcające dla każdego. A więc zamiast strzału w dziesiątkę, były to trafienia typu kulą w płot.

Wreszcie na koniec moje własne refleksje:

- *Tam gdzie są duże pieniądze czasem pozostaje trochę serca* (wątek filantropijny Woz, Gates)
- *Trudno do końca pozostać na piedestale, nawet jak się stworzyło wszystko od początku* (przypadek Kena Olsena w firmie DEC-Digital), gdyż interes pozostaje interesem.
- *Czasem bezinteresowność staje się "opłacalna"* (przypadek Linusa Torvaldsa)

DODATEK 3

KRÓTKA HISTORIA PRZEMYSŁU KOMPUTEROWEGO

Krótka historia przemysłu komputerowego - Narodziny i upadki

Opowiadam o przemyśle komputerowym jako świadek naoczny, gdyż towarzyszę informatyce od ponad 50 lat, najpierw jako student "computer science" a potem jako analityk i programista, a w koncu jako tzw. "knowledge worker". Wiele przemian w tej dziedzinie miało bezpośredni wpływ na to co robiłem i w czym brałem udział. W minionych latach nie wszystko było proste. Komputery i języki programowania oraz bazy danych zmieniały się tak często, że – nawet doszkalając się bez przerwy - trudno było nadążyć z wiedzą i umiejętnościami. Obecne zmiany w porównaniu z tamtymi są doprawdy niewielkie. Wtedy co parę lat zmieniały się całkowicie generacje maszyn. Trzeba było zaczynać ciągle prawie od zera. Szczególnie dotyczy to programistów używających języków maszynowych niskiego poziomu (assemblera lub symbolicznych), oraz personelu obsługi komputerów (ze względu na nowe systemy operacyjne). Z perspektywy lat tego nie widać. To, co było, można teraz odczytywać jako ciekawostki lub relikty przeszłości. Firmy komputerowe - tak jak i ludzie - nie były oszczędzane. Rodziły się i znikały, przeważnie bez rozgłosu. Czasem ich losy można śledzić jak akcję w powieści kryminalnej. Do "złośliwości" historii zaliczyć można prawie całkowite przemilczanie długiej -bo liczącej prawie 100 lat- epoki maszyn licząco-analitycznych (opartych na technologii kart dziurkowanych), na barkach których z powodzeniem realizowano powszechnie masowe przetwarzanie danych, począwszy od gospodarki materiałowej skończywszy na spisach powszechnych a nawet obsłudze wyborów prezydenckich.

Jednym z typowych zjawisk w przemyśle komputerowym było zanikanie firm (marek) lub działów komputerowych w firmach pionierskich, które najbardziej przyczyniły się do rozwoju przemysłu komputerowego. Chodzi o tak znane firmy jak Remington-Sperry-Rand, Control Data Corporation, Digital Equipment Corporation (pomijając takich pomniejszych producentów jak Ferranti, General Electric, Honeywell-Bull, ICL, RCA, Burroughs) . Kto wie jak wyglądałaby dzisiaj informatyka, gdyby potencjał i pomysły twórcze tkwiące w tych firmach zostały spożytkowane? A może można zaryzykować twierdzenie, że "lepsze jest wypierane przez gorsze (czytaj tańsze)" albo że firmy łożące (zbyt) dużo na prace badawczo-rozwojowe uwierzyły w twierdzenie, że "lepsze wypiera gorsze" ?

Uchowała się najpotężniejsza firma - IBM - która kiedyś słynęła z produkcji wspomnianych już analitycznych maszyn liczących (a na początku XX w. jako Corporation Computing Tabulating Record produkowała wagi przemysłowe, krajalnice do sera i wędlin oraz maszyny rejestracyjne), a potem utrwalony został jej wizerunek jako producenta superkomputerów i dużych komputerów (mainframes serii 360, 370, 390). Firma ta znana jest też z tego, że w 1955 r wyprodukowała 5MB dysk i w 1980 r wypuściła na rynek pierwsze komputery osobiste. O wielkości firmy IBM świadczy nie

tylko kapitał (ponad 300 związków kapitałowych i 20000 spółek, dochody z bardzo dużej liczby patentów), ale i stan zatrudnienia (kiedyś ponad pół miliona osób, a w 2000r ponad 300 000 osób).

Na etapie wczesnego rozwoju przemysłu komputerowego, kiedy raz po raz dokonywano kolejnych wynalazków (pamięci ferrytowe, tranzystory, układy scalone, taśmy magnetyczne, dyski itp.) prym wiodły firmy posiadające kapitał, początkowo powiązane z ośrodkami badawczymi na uczelniach, a potem same łożące na prace rozwojowe (np. IBM rocznie wydawał 5-6 mld dolarów). Wszystko było wówczas w powijakach: brak wydajnych systemów operacyjnych, oprogramowania narzędziowego i odpowiednich języków programowania. Nie było wówczas jeszcze standardów międzynarodowych. Każda seria komputerów "żyła w swoim własnym świecie", notabene bardzo skomplikowanym, skoro sprytne "podsunięcie" (prawdopodobnie) krajom socjalistycznym serii IBM/360 do skopiowania na postać tzw. maszyn jednolitego systemu RIAD, zajęło im tyle lat, że popadły w opóźnienie technologiczne nie do odrobienia, gdyż świat zachodni poszedł w tym czasie do przodu z nowszą technologią.

W tej części publikacji piszemy głównie o firmach. Nie wolno jednak zapomnieć, że podwaliny konstrukcyjne komputerów stworzone zostały w czasie i po II wojnie światowej głównie na uczelniach amerykańskich oraz angielskich. Mamy tutaj na myśli raporty o architekturze komputerów autorstwa von Neumanna w Princeton (komputer składa się z pamięci, jednostki centralnej, urządzeń wejścia-wyjścia, program i dane przechowywane w pamięci komputera - a nie wczytywane każdorazowo), praktyczne osiągnięcia Eckerta i Mauchly'ego w Pensylwanii (komputer ENIAC) oraz Howarda Aikena w Harvardzie. Dorobek ten wchłaniany był potem przez przemysł, często wraz z ludźmi (np. przejście Eckerta i Mauchly'ego do Remington Rand dały początek działowi komputerów komercyjnych UNIVAC w tej firmie). W W.Brytanii poważny wkład wnieśli m.i. Maurice Wilkes i Alan Turing. Nie można zapomnieć o zasługach Konrada Zuse'go w Niemczech, ale jego wpływ na światowy przemysł komputerowy był niewielki, gdyż pracował samotnie w Niemczech.

Pierwsze komputery do przetwarzania danych

Zastosowanie komputerów do zarządzania nie było oczywiste na początku przemysłu komputerowego. Komputery służyły głównie do obliczeń naukowych na uczelniach lub w wojsku. Opiszemy poniżej jak doszło do wyprodukowania pierwszych komputerów do tego celu.

Firmy komputerowe przeważnie zakładane były przez prężne utalentowane jednostki, ale casus firmy Lyons świadczy o tym, że na wczesnym etapie przemysłu komputerowego również w ramach każdej większej firmy - niezależnie od jej branży - mając wsparcie naukowe konsultantów można było rozpocząć produkcję komputerów. Decydujące znaczenie w tym przypadku miała mądrość kierownictwa firmy, w szczególności wyczucie nadchodzącego postępu technicznego i biznesowego. Do produkcji na początku nie potrzeba było znacznego kapitału, a własne prace badawcze zastępowane były przez konsultacje naukowców posiadających dorobek w tej dziedzinie.

W 1951 roku brytyjska firma gastronomiczna J. Lyons and Co. "zbudowała" dla własnych potrzeb duży jak na owe czasy **komputer LEO** (Lyons Electronic Office) - pierwszy europejski komputer do przetwarzania danych. Obsługiwał kilkaset restauracji, piekarń i kawiarń w zakresie ewidencji sprzedaży, badania popytu (mające na celu zmniejszenie zapasów magazynowych), obliczania płac itp. Maszynę oparto o rozwiązania konstrukcyjne komputera EDSAC (Electronic Delay Storage Automatic

Calculator), zbudowanego pod kierownictwem Maurice Wilkesa w Cambridge w Anglii oparciu o ideę von Neumanna. Komputer wybudowano kosztem ok.75 tys. funtów w oparciu o konsultacje naukowców z tej uczelni. Głównym konstruktorem komputera był Dr. John Pinkerton. Zasługą firmy było to, że pierwsza dostrzegła możliwość użycia komputerów do zadań administracyjno-biznesowych, gdyż wszystkie dotychczasowe ich zastosowania dotyczyły obliczeń naukowych. Firma ta była również prekursorem outsourcingu, gdyż LEO wykonywał obliczanie płac również dla innych firm (m.i. FORDa UK).

Zapotrzebowanie na te komputery było tak duże, że Lyons utworzył firmę LEO Computers Ltd., która rozpoczęła seryjna produkcję komputerów. Komputery LEO III pracowały aż do roku 1981 (m.i w British Telecom). Komputery LEO dopasowane były do potrzeb tzw. przetwarzania danych m.i. dzięki temu, że posiadały rozbudowany system wejścia-wyjścia (perforowane taśmy i karty, drukarka wierszowa, równoczesna obsługa wielu buforów wejścia-wyjścia). Późniejsze wersje tego komputera miały zaawansowany system operacyjny Master, obsługujący równocześnie do 16 programów. Firma LEO Computers przechodziła różne koleje losu i w końcu jako English Electric LEO Computers Ltd włączona została do ICT {International Computers and Tabulators}.

Skoro piszemy o pierwszym europejskim komputerze do przetwarzania danych administracyjno-biznesowych, warto opisać to w jaki sposób w tym samym czasie powstał podobny pierwszy amerykański komputer.

Pionierem komputerów do przetwarzania danych biznesowych była firma Remington Rand, znana z produkcji maszyn do pisania, urządzeń magazynowych oraz maszyn do szycia i elektrycznych maszynek do golenia. Na potwierdzenie tego, przytoczymy mało znany fakt, że już w 1949 roku wyprodukowała ona pierwszy w świecie komputer biznesowy komputer 409, notabene sprzedany potem do Japonii gdzie był w ogóle pierwszym zainstalowanym komputerem. Komputer ten był wytwarzany jeszcze potem jako Univac 60/120, ale firma miała większe ambicje, gdyż w 1950 roku przejęła chyba najbardziej wówczas naukowo i doświadczalnie zaawansowaną firmę EMCC (Eckert-Mauchly Computer Corporation), założoną przez twórców pierwszej elektronicznej maszyny liczącej ENIAC w latach 1943-1946 r. na uniwersytecie w Pensylwanii. To właśnie naukowcy Eckert i Mauchly byli w Remington R. konstruktorami pierwszego amerykańskiego komercyjnego komputera **UNIVAC** (UNIVersal Automatic Computer), zamówionego w 1948 roku w EMCC przez amerykańskie Biuro Spisu Ludności. Pierwsze dwa egzemplarze tego komputera pracowały dla tego biura. Na piątym z nich, zbudowanym dla Komisji Energii Atomowej, z powodzeniem przewidziano (na podstawie 1% próbki) zwycięstwo Eisenhowera w wyborach prezydenckich. Warto w tym miejscu odnotować dalsze losy firmy Remington Rand. W 1955 r połączyła się ze Sperry (w 1965 r: pierwszy wieloprocesorowy komputer 1108), tworząc Sperry Rand, a 30 lat później z połączenia tejże firmy z zasłużoną firmą komputerową Burroughs (1961r: B5000 - pierwszy komputer dwuprocesorowy z wirtualną pamięcią, 1959r: pierwszy czytnik pisma MICR) powstał Unisys.

Przykłady komputerów LEO i Univac świadczą o tym, że już stosunkowo wcześnie dla znaleziony został szeroki zakres zastosowań, wykraczający poza obliczenia naukowe.

Komputery te miały jeszcze taką zasługę historyczną, że zapoczątkowały koniec epoki maszyn analitycznych (tabulatorów, sorterów, dziurkarek, sprawdzarek), dominujących w dziedzinie przetwarzania danych administracyjno-ekonomicznych

Błędy w strategiach rozwojowych

Nawet duże pieniądze na prace rozwojowe nie oznaczały, że firmy nie popełniały błędów w strategii rozwojowej.

Największa firma w dziedzinie "wszelakich" maszyn liczących - **IBM** - od początku była blisko naukowców i komputerów.

Już w latach 40-tych finansowała projekty w Harvard University dr Howarda Aikena,

który zafascynowany XIX- wiecznymi ideami Ch.Babbage'a, zbudował serię komputerów MARK opatrzoną przydomkiem Harvard. MARK I był przekaźnikowy, ale MARK IV już całkowicie elektroniczny, wyposażony w ferrytową pamięć operacyjną i w bęben magnetyczny .

W latach 50-tych IBM docenił walory Johna Backusa - absolwenta matematyki w Columbia University, który w 1957 r opracował pierwszy język wysokiego poziomu FORTRAN, a w 1959 roku wymyślił wspólnie z Naurem notację BNF (Backus Naur Form) do opisu składni języków tej klasy.

Natomiast w latach 70-tych IBM zwlekał długo z wykorzystaniem rewelacyjnej (a chyba nawet genialnej jak na owe czasy) pracy "A Relational Model of Data for Large Shared Data Banks", swojego pracownika Edgara Franka Codda, w której ten przedstawił model relacyjnej bazy danych. Skorzystał na tym Larry Ellison - twórca firmy ORACLE - który szybciej implementował idee Codda (w 1979 r powstała 1sza komercyjna relacyjna baza ORACLE, a dopiero 5 lat później IBMowska baza DB2).

Nie było to jedynie przeoczenie IBMu. Na przełomie lat 70 i 80-tych w laboratoriach IBMu nad technologią RISC i architekturą klient/serwer pracował Joel Birnbaum. Firma jednak- zaangażowana zbytnio w technologię mainframe'ów - zrezygnowała ze współpracy z Birnbaumem, twierdząc, że środowisko otwarte to zabawa dla studentów.

Błędy takie popełniał nie tylko IBM. W 1976 r. kierownictwo **Hewlett Packard** nie doceniło pomysłu swego pracownika Steve'a Wozniaka, który skonstruował prototyp komputera osobistego i zaoferował projekt pracodawcy. Niedługo potem Wozniak wraz ze Steve'em Jobsem założyli własną firmę -Apple Computer. Błąd swój HP niejako naprawił kilka lat później - namówiony przez odrzuconego w IBM twórcy technologii RISC Joela Birnbauma - inwestując 1 mln USD w rozwój tej technologii, dzięki czemu stał się czołowym producentem serwerów.

Różne przykłady dobrej strategii

Inwestowanie w prace rozwojowe nie jest jednak konieczne. Firma Dell Computer nie rozwija własnych technologii, a więc nie łozy zbyt wiele na prace rozwojowo-badawcze. Założyciel firmy - **Michael Dell** - zdecydował się na stosowanie uznanych standardów i kupowanie rozwiązań innych producentów. Zastosowano jeszcze inne podejście, które obniża ceny – dostarczano sprzęt bezpośrednio do użytkowników – a więc pominięto marże pośredników. A efekt ? W 1997 roku firma sprzedała swój 10 milionowy komputer, a w 2000 roku zajęła 1sze miejsce w świecie jako dostawca stacji roboczych. Kiedy innych dotykało bankructwo, firma pięła się w górę. Dynamika wzrostu została zahamowana w ostatnich latach, kiedy firma zaczęła mieć problemy z serwisowaniem klientów (m.i. przyczyniły się do tego słynne kłopoty z bateriami Sony do notebooków) a firmy HP, Apple i Lenovo zintensyfikowały działania na rynku.

Przykładem dobrej strategii rozwoju wydaje się być powstała w 1982 firma **SUN** Microsystems, która umiała trafić w trendy rozwojowe (architektura RISC, symetryczne wieloprzetwarzanie, technologia Javy) i znalazła dla siebie miejsce – swoją niszę rynkową – produkując komputery działające pod UNIXem (własny system operacyjny

Solaris) w środowisku sieciowym (stacje Sparc). SUN umiał również nawiązać alianse partnerskie z czołowymi producentami oprogramowania bazodanowego (Informix, Ingres, Oracle, Sybase) poszerzając w ten sposób potencjał zastosowań swoich komputerów. SUN produkował sprzęt oraz nie zrezygnował z prac rozwojowych. Można powiedzieć, że dzięki otwartości na to co dzieje się w świecie informatyki, SUNowi udało się to, na czym potknął się Digital. Można więc wyrazić uznanie dla czwórki założycieli tej firmy (Scott McNealy, Vinod Khosla, Bill Joy, Andy Bechtolsheim). Już na początku działalności (1983) zdobyli niezbędny kapitał dla działalności firmy zawierając z Computervision umowę OEM na 40 milionów dolarów. Wtedy kapitał zdobyć było trudniej - teraz typową drogą dla firmy w kategorii HighTech jest odniesienie pewnych sukcesów rynkowych, potem debiut giełdowy, emisja dużej puli akcji giełdowych i zwykle przejęcie większej firmy z dorobkiem.

Przykładem "autorskiej" (indywidualnej) drogi innowacyjnej w przemyśle komputerowym jest firma **Cray Research** założona w 1972 r i prowadzona do 1994 r przez Seymoura Cray'a, który najpierw pracował nad dużymi komputerami CDC w Control Data (której to firmy był współzałożycielem), a potem produkował już we własnej firmie superkomputery Cray, piastując tam prawie wszystkie możliwe stanowiska kierownicze. Kiedy uświadomił sobie potencjał mikroprocesorów założył następną firmę autorską SRC Computers, Niestety nie zdążył wykorzystać nowych technologii, gdyż zmarł w wieku 71 lat w wyniku obrażeń odniesionych w wypadku samochodowym w 1996 r w Colorado Springs.

O sukcesy w ostatnich 20 latach XX wieku było trudno

Nawet giganci mieli wówczas problemy. Na przykład w latach 80-tych firmy japońskie Fujitsu i Hitachi przypuściły ostry atak na **IBM** szybciej opracowując nowe produkty dla serii 370. Na rynku minikomputerów AS/400 walczył z produktami Sun Microsystems i Hewlett-Packard. Pogarszała się sytuacja IBM na rynku PC, gdzie agresywny marketing prowadziły m.in. Compaq i Dell. Po roku 1993, w którym poniósł poważne straty finansowe, IBM poza zwolnieniami personelu wprowadził poważne zmiany konstrukcyjno-technologiczne np. zmiana układów ECL na CMOS w mainframe'acch, dzięki czemu ich cena spadła i stały się one bardziej konkurencyjne.

Duże perypetie przeżywała firma pionierska firma **NCR** (1957- NCR304 pierwszy całkowicie tranzystorowy komputer do zastosowań w biznesie, 1968 - John L. Janning z NCR wynalazł liquid crystal displays LCD, 1974 – pierwszy czytnik kodów kreskowych), ponadto jeden z największych producentów bankomatów, dostawca software'u bazodanowego Teradata). W 1991 r firma NCR została przejęta przez AT&T, aby w 1998 roku z powrotem pojawić się jako niezależna firma.

Rywalizacji nie wytrzymała firma **Digital** (DEC), od początku kładąca nacisk na prace rozwojowe, zarówno w konstrukcji komputerów jak i oprogramowaniu, przyczyniając się w znacznym stopniu do rozwoju systemów operacyjnych (w tym do Unixa - poza własnym systemem OpenVMS) i takich urządzeń jak router, klastry komputerow oraz protokół ethernetu. Poważne kłopoty zaczęły się w połowie lat 90-tych. Poniosła ona w 1994 roku straty sięgające 2 mld dolarów, zaś w roku następnym (zapewne dzięki rozpoczęciu produkcji procesorów ALPHA) zdołała wygospodarować zysk 122 ml. Mimo tego jeszcze w tym roku Digital zwolnił 16000 osób a nastepne jego działania sprawiały wrażenie "wyprzedaży" firmy (np. sprzedaż Oracle'owi działu RdB i Data Repository, sprzedaż do CA systemu POLYCENTER do zarządzania środowiskiem sieciowym, potem sprzedaż firmie BEA Systems produktów ORB, MOM, ObjectBroker i DECmessageQ). Przyczyną nawrotu strat mogła być - niezależnie od braku strategii rozwojowej firmy -

ogólnoświatowa tendencja spadku opłacalności produkcji komputerów osobistych, które też zaczął produkować Digital. O trudnej sytuacji firmy świadczy fakt, że w październiku 1997 r. Digital sprzedał firmie Intel swój sztandarowy dział procesorów (produkujący m.i. Alphy). Finał nastąpił o północy czasu nowojorskiego z 25 na 26 stycznia 1998 r. kiedy to firma ta została przejęta za 9,6 mld dolarów (wg późniejszej wyceny suma ta uległa zmniejszeniu o 1 miliard) przez Compaq Computer. W dniu 12 czerwca 1998 r. na walnym zgromadzeniu akcjonariuszy nastąpiło oficjalne połączenie (a właściwie przejęcie) firmy Digital Equipment Corp. i Compaq Computer Corp.

Firma **Compaq** dokonała imponującego skoku, gdyż założona została dopiero w 1982 roku przez dwóch byłych pracowników Texas Instruments, a potem stała się drugą (po IBM) lub trzecią (IBM, HP) największą firmą komputerową na świecie. Tak więc młoda lecz ambitna i agresywna firma połknęła doświadczonego pioniera przemysłu komputerowego. Warto dodać, że w 1997 roku Compaq przejął (za 3 mld dolarów) inną znaczącą firmę TANDEM. We wrześniu 2001 roku firma Compaq została przejęta przez HP (za 20 mld dolarów zapłaconych w akcjach), co doprowadziło do utworzenia firmy o obrocie (87 mld USD) porównywalnym z IBM (88,4 mld w 2000r.)

Sukcesy firmy Dell i wchłonięcie fimy Digital przez agresywnego Compaqa w 1998 roku zamykały jakby dotychczasowy – pionierski rzec można - model rozwoju firm komputerowych. Rozwojem przemysłu komputerowego zaczęły rządzić twarde prawa rynkowo-kapitałowe.

Kulisy zakładania firm komputerowych jakby mieściły się w amerykańskim modelu "od czyścibuta do milionera". Do założenia firm, które obracają obecnie miliardami dolarów, kiedyś wystarczało mieć inwencję, pomysł biznesowy oraz zainwestować tysiąc - a czasem tylko kilkaset – dolarów. Przykładów nie brakuje.

- Firmę **HP** założyli w garażu z kapitałem 538 USD w 1938 roku studenci David Packard i William Hewlett. Głównym zadaniem firmy początkowo była produkcja oscylatorów dźwięku dla Walta Disney'a. Obraca ona teraz miliardami dolarów i zatrudnia ponad 100 tysięcy osób.

- Również w garażu założona została w 1976 roku firma **Apple** (produkująca komputery Mackintosh) przez dwóch Steve'ów : Wozniaka i Jobsa.

- Podobnie wygląda sprawa z firmą **Digital** utworzoną - przez Kena Olsena -w starej przędzalni w 1957 roku z kapitałem 70 tysięcy dolarów - 30 lat później firma była wyceniana na 14 mld dolarów

- W 1977 roku Larry Elison wraz z Robertem Minerem założyl firmę **Oracle** z kapitałem założycielskim wynoszącym zaledwie 1200 USD. W 2001 roku wartość rynkowa firmy wynosiła 86 mld dolarów. Majątek Elisona oceniany był w 2001 roku na 26 mld dolarów

- Twórcy **Microsoftu** B.Gates i Paul Allen w 1974 roku kupili jeden z pierwszych mikroprocesorów 8008, a w 1975 roku zostali zatrudnieni przez firmę MITS (Micro Instrumentation and Telemetry Systems) do opracowania interpretera języka BASIC dla komputera Altair z procesorem 8080. W 1976 roku odeszli z tej firmy i zajęli się własnym biznesem, m.i. prowadzili prace nad systemem operacyjnym MS DOS (ukazał się w 1981 roku) dla IBM'owskiego prapeceta z procesorem 8088. W 1983 roku rozpoczęli (m.i. pod wpływem interfejsu

graficznego firmy Apple) linię systemów operacyjnych Windows. W 2001 roku wartość rynkowa firmy Microsoft wynosiła 369 mld dolarów. W 2001 roku majątek B.Gatesa oceniany był na 54 mld dolarów, a w 2002 r na 43 mld . Gates jest najbogatszym człowiekiem Ameryki, zaś Paul Allen znajduje się w 1szej piątce.

- Firmę **Dell Computer** z kapitałem ok.1 tys. dolarów założył w 1984 roku eks-student (porzucił studia po 1 roku) Michael S. Dell. Działając jako prezes i dyrektor wykonawczy doprowadził wartość rynkową firmy do 50-60 mld dolarów

- Firma **Intel** została założona w 1968 r.- przy bardzo niskim kapitale własnym - przez grupę inżynierow (w tym ojca pierwszego układu scalonego Bob Noyce'a i współtwórcę tranzystora Williama Shockleya), którzy porzucili firmy Fairchild Semiconductor i Bell

- Firma **Compaq Computer** została założona w 1982 r. przez 3 (Rod Canion, Jim Harris i Bill Murto) byłych pracowników Texas Instrument. Nazwa pochodzi od zbitki angielskich słów - compatibility i quality.

- Firmę **SAS** (Statistical Analysis System) Institute Inc. założył w 1976 r. Jim Goodnigt wraz z 3 współpracownikami. Biegły w programowaniu mainframe'ów IBM, odszedł z miejscowego uniwersytetu aby już na własny rachunek realizować zlecenia na obliczenia statystyczne od stanowej administracji rolnej. Po 20 latach firma operowała kapitałem 420 milionów dolarow i zatrudniała 3200 osob (w tym 2100 w USA), pozostając cały czas w tych samych prywatnych rękach - firma całkowicie prywatna (niegiełdowa). W 2005 roku dochody firmy sięgały ponad 1,5 mld dolarów.

Posumowanie

Pierwsze komputery powstawały w najrozmaitszy sposób. Wytwarzano je nie tylko w poważnych korporacjach, ale też w garażach, a nawet firmie gastronomicznej.

Podstawy teoretyczne o zasięgu międzynarodowym wraz z egzemplarzami komputerów, w których je wdrożono - zawdzięczamy kilku uczelniom (głównie amerykańskim i angielskim). Na uwagę zasługuje również indywidualny dorobek Konrada Zuse w Niemczech. Rozwój przemysłu komputerowego początkowo odbywał się w dużych firmach produkujących przedtem inne produkty, a potem nabrał radykalnego przyśpieszenia dzięki inwencji kilku młodych prężnych i ambitnych osób.

Rozwój zastosowań przerósł najśmielsze oczekiwania. Kiedyś myślano, że już kilka większych komputerów zaspokoi potrzeby wszystkich wielkich koncernów amerykańskich, a teraz kilkadziesiąt milionów pracuje w każdym większym kraju. Ba, świat cały opleciony jest komputerową siecią internetową, z której korzystamy wszyscy - od dzieci do staruszków. Bez komputerów niemożliwe jest funkcjonowanie gospodarki, a w szczególności biznesu bankowego (np. dzisiaj nie do wyobrażenia jest powrót do papierowych przelewów i przewożenia dokumentów do krajowej izby rozliczeniowej).