A. Mamuchiera

40 LAT INFORMATYKI W POLSCE

Warszawa, październik 1988

Antoni Mazurkiewicz

Jak się programowało XYZ
czyli początki programowania w Polsce.

Były to czasy gdy nie tylko nie było w Polsce komputerów, ale nawet słowo to nie istniało. Były natomiast 40 lat temu zamierzenia, które wielu wydawały się mrzonkami: czy rzeczywiście można w kraju zbudować "mózg elektronowy"? Jest dowodem dalekowzroczności dyrektora ówczwsnego Państwowego Instytutu Matematycznego, profesora Kuratowskiego, że zawierzył on młodym inżynierom elektronikom: Bochenkowi, Łukaszewiczowi i Marczyńskiemu, dając im szanse przekonania niedowiarków.

O historii rozwoju sprzętu liczącego, ówczesnej technice i technologii mowa jest w innym miejscu niniejszego tomu. Ja natomiast pragnę przypomnieć jak wyglądała praca nad oprogramowaniem pierwszych polskich maszyn.

Mówiąc o maszynie XYZ będę miał na myśli również jej następczynie, maszyny ZAM-2, Alfa, Beta i Gamma, dla których XYZ była modelem użytkowym.

Programować zaczęliśmy abstrakcyjnie, bez maszyny i bez jakichkolwiek doświadczeń praktycznych w tej dziedzinie. Początkowo jedynie Andrzej Wakulicz i Adam Empacher wiedzieli co to jest elektroniczna maszyna cyfrowa i na czym polega jej programowanie, potem matematycy pracujący przy maszynach analogowych (Józef Winkowski, Tomasz Pietrzykowski i autor tego opracowania) dołączyli do wtajemniczonych. Żaden z nas nie widział wówczas działającej maszyny cyfrowej, wiedzę o programowaniu czerpaliśmy z nielicznych publikacji zagranicznych; pamiętam, że jedną z nich była książka Wilkesa z Wielkiej Brytanii. Było to jedyne źródło naszej wiedzy o kodach, adresach, pseudorozkazach, tworzeniu pętli i rozgałęzień. Konstrukcja prostych algorytmów i doprowadzenie ich do postaci programów było wówczas czynnością nieszablonową

i dostarczało nam niemało satysfakcji twórczej.

Okres ten to dominacja można rzec abstrakcyjnego programowania zagadnień głównie numerycznych, koncentracja na działaniu programów w pamięci maszyny, przy niemal całkowitym pomijaniu spraw dotyczących wprowadzania i wyprowadzania informacji.

Gdy tylko budowana maszyna cyfrowa XYZ zaczęła nabierać realności, gdy ustalono jej repertuar rozkazów i stały się znane jej podstawowe cechy, przystąpiono do tworzenia prawdziwego, realnego oprogramowania. W tym czasie dołączyli do nas między innymi Jan Borowiec, Krzysztof Moszyński, Jerzy Swianiewicz i Andrzej Wiśniewski, którzy w tej działalności odegrali istotną rolę. Naszym celem było utworzenie biblioteki programów (numerycznych, oczywiście, o przetwarzaniu danych jeszcze się wówczas nie śniło) i krótkiego systemu operacyjnego pozwalającego ładować programy binarne do pamięci i przemieszczać je w niej w ograniczonym zakresie. Oprogramowanie rosło, lecz nikt jeszcze nie wiedział jak będzie ono działało w praktyce, jak szybkie będzie liczenie, jak będzie wyglądać praca maszyny, jakie będą wyniki. Maszyna, chociaż już teoretycznie znana, była bowiem jeszcze tworem abstrakcyjnym. Czekaliśmy więc niecierpliwie na możliwość praktycznego potwierdzenia naszych abstrakcyjnych koncepcji.

I wreszcie pewnego dnia XYZ ruszył. Trzy stojaki z panelami wypełnionymi lampami, pulpit operatora i reproducer kart dziurkowanych wypełniały dość spore, specjalnie zaadoptowane pomieszczenie. Na pulpicie widać było kilka rzędów kluczy i dwa okrągłe oscyloskopy. Na jednym z nich można było obserwować zawartość wybranej "rury" pamięci (32 słowa czyli 64 bajty), oczywiście w postaci binarnej, zakodowanej przez jaśniejsze i ciemniejsze kropki, na drugim zawartość rejestrów akumulatora i mnożnika w postaci klasycznych ciągów impulsów. Oglądaliśmy z przejęciem wzrastanie zawartości liczników (wówczas dla nas zawrotnie szybkie, zmienność dopiero szóstego bitu od końca dawała się zauważyć! XYZ liczył bowiem z niebagatelną w tym czasie szybkością ok.1000 operacji arytmetycznych na sekundę). Na drugim oscyloskopie można było

zobaczyć na własne oczy jak powstaje wynik dodania, pomnożenia, a nawet podzielenia dwóch słów binarnych. W tym czasie charakterystyczny był w Zakładzie Aparatów Matematycznych widok programisty siedzącego przy pulpicie XYZ wpatrującego się w owe oscyloskopy i naciskającego jeden klucz, bardzo ważny i najczęściej używany, powodujący wykonanie pojedynczego kolejnego rozkazu programu (z angielska "single shot"). Tak właśnie uruchamoało się programy: wykonywało się mianowicie kolejno instrukcję po instrukcji i obserwowało się na oscyloskopie efekty ich działania. Pamięć XYZ (zbudowana na liniach opóźniających skonstruowanych z rur stalowych wypełnionych rtęcią, w których rozchodziły się fale akustyczne opóźniające bieg impulsów) składała się z 1024 słów 18 bitowych; proszę sobie wyobrazić, jak wielki wysiłek musiał być włożony w optymalizację przestrzenną programów i rządzącego nimi systemu operacyjnego, aby móc policzyć jakieś rzeczywiste zadanie! Najwięcej kłopotów było z wyprowadzaniem wyników. Początkowo jedynym medium wyjściowym były karty perforowane. Urządzenie wyjściowe dziurkujące karty było wielkości biurka, niezmiernie ciężkie, masywne i hałasujące tak, że wyprowadzanie wyników było słychać w całym gmachu przy Śniadeckich 8. Co więcej, nie było na miejscu urządzenia tabulującego zawartość kart, trzeba było jeździć z kartami do Głównego Urzędu Statystycznego aby dowiedzieć się, co maszyna naniosła na karty wyjściowe. Dopiero zainstalowanie wejścia i wyjścia na taśmie papierowej i zainstalowanie dalekopisów uczyniło sytuację znacznie wygodniejszą.

Na charakter oprogramowania maszyn rodziny XYZ wpłynęła w istotny sposób ich "słowowa" koncepcja. Jest to być może najpoważniejsza różnica jaka istnieje między architekturą XYZ, a współczesną "znakową" organizacją najpopularniejszych komputerów. Można to wytłumaczyć ówczesną dominacją przetwarzania numerycznego nad przetwarzaniem tekstowym, preferującym w oczywisty sposób koncepcję znakową. Można też widzieć w tym wpływ komputerów IBM serii 700, na których XYZ był wzorowany. Rzecz jasna, słowowy charakter maszyny, jawnie reprezentowany w języku adresów symbolicznych SAS (języku

assemblera XYZ), najwyraźniej uwidocznił się w programach operacyjnych i translatorach języków programowania, ale nie tylko. W jednoadresowych maszynach podobnych do XYZ słowo reprezentujące rozkaz podzielone jest w zasadzie na dwie części: jedną określającą kod rozkazu, i drugą, określającą jego argument (zazwyczaj jest nim adres w pamięci wewnętrznej). Pewna liczba pozostałych bitów jest na ogół na różnego typu znaczniki. Przy takiej organizacji zwiększenie repertuaru rozkazów wiąże się ze zmniejszeniem zakresu bezpośredniego adresowania i odwrotnie - chcąc powiększyć pamięć adresowaną bezpośrednio trzeba zmniejszyć liczbę dostępnych instrukcji. Ta współzależność była zapewne przyczyną braku takich możliwości maszyny, które znacznie ułatwiłyby nam pracę nad budową programów i usprawniłyby cały system programowania. Chodzi tu o umożliwienie stosowania różnorodnego typu argumentów instrukcji: argumentu natychmiastowego, (co pozwoliłoby znacznie skrócić i przyspieszyć wiele programów użytkowych, adresu pośredniego (co usprawniłoby znacznie komunikację między podprogramami i pozwolić by mogło na rozbudowywanie pamięci) i o wprowadzenie dodatkowego rejestru indeksowania, którego podstawowym celem byłoby usprawnienie gospodarki pamięcią. Trzeba jednak dodać gwoli sprawiedliwości, że dopiero teraz, po upływie lat, widać jasno zalety i wady takich, a nie innych rozwiązań architektonicznych; w czasie konstrukcji XYZ nie były one tak bardzo oczywiste.

Warto przypomnieć jakie były ówczesne metody programowania i uruchomiania programów. Oprogramowanie podstawowe XYZ było pisane w języku maszyny. Część operacyjna instrukcji miała swój literowy symbol, część adresową stanowiła liczba w systemie ósemkowym. Gdy nie było jeszcze systemu SAS wszystkie instrukcje programu były przez programistę przekształcane na postać dwójkową i przenoszone na karty ręcznie preforowane urządzeniem pozwalającym wyciąć w karcie jedynie pojedynczą dziurkę. Była to era powszechnego użycia tzw. "trafaretu", to jest odpowiednio pokolorowanej karty o maksymalnie wyciętych dziurkach. Kartę taką przykładało się do przygotowanej karty z

instrukcjami i sprawdzało się poprawność rozmieszczenia w niej dziurek. Pozwoliło to na względnie łatwe /?/ zorientowanie się w strukturze instrukcji zakodowanych na karcie. Poprawienie błędu polegało z reguły na wydziurkowaniu karty od nowa /to znaczy 12 binarnych instrukcji/.

Oprogramowanie zaczęło się od napisania tzw. "pętli samoładującej" - najprostrzego programu wciągającego - mieszczącego się na jednej karcie. Karta taka poprzedzała przeznaczony do wciągnięcia plik kart z programem i danymi. Wprowadzenie programu do pamięci nie było bezproblemowe: po dłuższym użytkowaniu karty puchły i zacinały się w podajniku urządzenia czytającego. Ale programistów trudno było zniechęcić: wynaleźli oni technikę usprawniania procesu wejścia polegającą na podpalaniu krytycznej krawędzi karty, co przywracało jej dostateczną cienkość i zdolność do przechodzenia przez szczelinę urządzenia czytającego. Mimo to, w tych warunkach uruchomienie programów trwało długo, a jego zakończenie było prawdziwym świętem dla twórcy programu.

Niewielka objętość pamięci i niezbyt wielka szybkość XYZ wywarły widoczny wpływ na metodologię programowania tej maszyny. Dążono do jak najmniejszej objętości programu jeśli przy tym udawało się zwiększyć jego szybkość, sukces był pełny. Złoty okres przeżywało programowanie trickowe: istniało nawet w folklorze programowym pojęcie katalogu chwytów programistycznych na każdą okazję. Technika programowania wynikała też ze skąpych możliwości dokonywania zmian w programach: na ogół programy były zbudowane z "epicykli": wyskoków do sekwencji brakujących początkowo instrukcji. Każdy program, można by rzec, był dziełem sztuki: nie ważny był stopień jego komplikacji, nieczytelność, trudność dokonywania zmian: istotne, czy był on objętościowo niewielki, a w miarę możliwości szybki. Rzecz jasna, opieką nad tak napisanym programem mógł sprawować wyłącznie jego autor.

Nawyki programistów wyrobione w tym "heroicznym" okresie programowania, zemściły się później przy oprogramowywaniu maszyn serii ZAM 31, ZAM 41 i pokrewnych, maszyn większych, o bogatszych możliwościach, gdy należało pielęgnować duże

programy. W czasach XYZ trudno było mówić o jakiejś metodologii programowania, czy programowaniu systematycznym. A jednak....Programiści chcąc usystematyzować i ułatwić sobie proces programowania przyjmowali pewne zasady, które w istocie stanowiły zaczątek programowania modularnego i strukturalnego. Jeden z najlepszych programistów jakich znałem Stefan Sawicki, na długo przed pojawieniem się informacji o modularnym programowaniu, programował zasadę: "dobry program powinien zmieścić się na jednej kartce papieru zeszytowego". Oczywiście, program taki składał się na ogół z wielu podprogramów, z których każdy również musiał mieścić się na jednej kartce papieru. Często okazywało się, że programy zbudowane według zasady Sawickiego nie były istotnie wolniejsze od trickowych, ani też od nich o wiele dłuższe.

Język SAS stał się podstawą dalszego oprogramowania maszyn rodziny XYZ. Należy stwierdzić wyraźnie, że oprogramowanie to zawdzięczało swój sukces starannemu projektowi systemu operacyjnego i języka SAS. Bez gruntownego przemyślenia tych konstrukcji nie powstałoby ani SAKO ani nie stworzono by biblioteki programów współpracujących z tym system. Nie oznacza to rzecz jasna doskonałości SAS. Niedogodnościami użytkowymi tego języka nie należy jednak obciążać jego projektantów; wynikały one raczej z pewnych niedoskonałości organizacji maszyny o czym już wspomniano wyżej (brak argumentu natychmiastowego, pośredniego adresowania, braku rejestru indeksowania; rejestr taki pojawił się dopiero w serii ZAM2). Natomiast cała konstrukcja "assemblera" i sposób organizacji pamięci były rozwiązane w sposób niemal współczesny. Słowo "niemal" użyte jest tutaj ze względu na drugą różnicę jaka dzieli architekturę maszyny XYZ od standardów współczesnych; brak pamięci typu "stos". Trzeba to wspomnieć, że podstawy teoretyczne pamięci stosowej dał Pawlak w swym projekcie maszyn bezadresowych wyprzedzając znacznie algorytm Samelsona i Bauera wyliczania wartości (lub kompilowania) wyrażeń arytmetycznych, pracą Dijkstry o rekursywnym programowaniu oraz pierwszą praktyczną realizację pamięci stosowej przez firmę Burroughgs.

Przełomu w metodologii programowania dokonał język SAKO (system automatycznego kodowania) po pierwsze uświadomił on programistom, że programy mogą być zbudowane ze standardowych części i że ich optymalność czasowa (poza być może oprogramowaniem systemowym) nie jest tak ważna jak początkowo sądzono. Po drugie system ten dał narzędzie, które pozwoliło zapomnieć o licznych detalach programu tak skrupulatnie analizowanych w programowaniu trickowym, natomiast zwróciło uwagę na ogólną stukturę programów, sposób globalnej gospodarki pamięcią, przekazywanie informacji wyodrębnionymi częściami dużych programów. Początkowo wytrawni programiści stronili od SAKO, pozostawiając programowanie w tym systemie nowicjuszom i leniuchom. Ale wkrótce okazało się, że ci nowicjusze i te leniuchy uruchamiają swe programy wielokrotnie szybciej niż doświadczeni konserwatyści. Okazało się też, że o to chodzi, że mniej więcej 10 procentowe wydłużenie czasu działania programu nie ma, w większości zastosowań, istotnego znaczenia. Ma natomiast znaczenie całkowity czas uzyskania odpowiedzi i czas potrzebny na niemal nieuniknione modyfikacje programu. Nie ma potrzeby dodawać, że modyfikacja programu w SAKO była nieporównanie łatwiejsza od tejże w języku symbolicznym.

Sako było językiem wyższego poziomu nazywanym często "polskim Fortranem". Nic zresztą dziwnego, skoro SAKO było wzorowane na tym języku. Liczne udoskonalenia i zmiany w stosunku do Fortranu uczyniły jednak z SAKO język nowy i oryginalny. Kompilator SAKO był dwuprzebiegowy: w pierwszym przebiegu zwroty SAKO zamieniane były na ciągi rozkazów SAS, które w drugim przebiegu były łącznie zbierane i adresowane. Skompilowany program mógł być w postaci tzw. taśmy binarnej zdolnej do wprowadzenia do pamięci maszyny już bez dodatkowych zabiegów. Program w SAKO mógł składać się z niezależnie kompilowanych modułów. Podobnie jak i na SASie swoiste piętno na SAKO wycisnęła architektura maszyny, chociaż - co zrozumiałe - w znacznie mniejszym stopniu.

Przypomnijmy, że pamięć wewnętrzna maszyn rodziny XYZ zawierała 1024 słowa (wg dzisiejszej terminologii ok. 2kB) .

SAKO projektowano dla maszyny wzbogaconej o pamięć zewnętrzną (bębnową) o pojemności 16.000 słów (ok. 32kB), i częstotliwości 1,25kHz. Przy tych parametrach zdecydowano się (słusznie jak się wkrótce okazało) na działania stałoprzecinkowe (podprogramy zmiennoprzecinkowe zużywałyby zbyt wiele czasu), metodę kompilacyjną (programy interpretujące zajęłyby zbyt dużo miejsca) i jawne stronicowanie pamięci (w postaci tzw. "rozdziałów" co umożliwiało programistom na panowanie nad wymianą stronic i pozwalało uniknąć wprowadzania dość skomplikowanych procedur systemu operacyjnego). Na kształt języka SAKO wpłynęła również stałoprzecinkowość maszyny. Na interpretacyjne wykonywanie działań zmiennoprzecinkowych nie mogliśmy sobie pozwolić z uwagi na małą pojemność pamięci wewnętrznej XYZ i na niezbyt dużą jej szybkość. Rozwiązaniem było więc umożliwienie liczenia w stałym przecinku przez wprowadzenie do języka SAKO deklaracji skalujących określających położenie przecinka pozycyjnego w liczbach ułamkowych. Podział na liczby całkowite i ułamkowe (skalowane) oszczędzał pamięć i przyspieszał liczenie.

Istotną nowością w stosunku do Fortranu i innych współczesnych języków programowania wysokiego poziomu były wyrażenia Boole'owskie, w których wartości zmiennych interpretowane były jako 36-elementowe ciągi zero-jedynkowe. Operacje arytmetyczne miały w tych wyrażeniach zmienione znaczenia, np. mnożenie oznaczało koniunkcję słów (bit po bicie). Dzięki takim wyrażeniom można było w SAKO operować na dowolnych częściach słowa w postaci binarnej, wycinać lub maskować pewne grupy bitów, porównywać je i zastępować przez inne. Wypada podkreślić, że wszystko to można było programować w języku wysokiego poziomu, co długo jeszcze nie było możliwe w innych językach. Drugą przewagą nad Fortranem był system procedur, co prawda jeszcze nie rekursywnych, ale umożliwiających pewną prostą postać podstawienia przez nazwę (podstawienie przez referencję do zmiennych tablicowych i nazw procedur) i również prostą ale skuteczną metodę przesyłania parametrów tranzytowych ("by passing parameters"). Cechy te,

których nie sposób szczegółowo objaśnić w tym artykule, pozwalały na wygodne tworzenie biblioteki podprogramów w SAKO i ich łatwe włączanie do pracy w innych programach użytkowych.

Jak ocenić ówczesne systemy programowania? Bez fałszywej skromności można je uznać za dobre, a nawet bardzo dobre, biorąc oczywiście pod uwagę dostępny na tamten czas sprzęt i ogólnoświatowy poziom informatyki lat pięćdziesiątych. Z perspektywy trzydziestu lat jest nawet niejako zaskoczeniem dla współczesnego informatyka jak aktualna była koncepcja języka adresów symbolicznych SAS (jak współcześnie mówimy języka "assemblera"), systemu operacyjnego XYZ, czy języka SAKO. Należy w tym miejscu podkreślić rolę Leona Łukaszewicza, który zachęcił programistów do podjęcia ambitnych zadań, wyglądających dla wielu z nas mało realnie; był on również autorem wielu koncepcji dotyczących tworzonego oprogramowania. W skład zespołu oprogramowującego XYZ wchodzili (przynajmniej czasowo): Elga Adamian, Jan Borowiec, Andrzej Brzostowski, Zbigniew Bzymek, Ludwik Czaja, Jowita Koncewicz, Danuta Kosecka, Jadwiga Lewkowicz, Leopold Łabanowski, Maria Łącka, Antoni Mazurkiewicz, Iwona Messner, Jacek Moszczyński, Krzysztof Moszyński, Jerzy Mysior, Ryszard Okrasiński, Barbara Pallasch, Elżbieta Pleszczyńska, Henryk Radzikowski, Apolonia Tymińska, Roman Rędziejowski, Andrzej Salwicki, Stefan Sawicki, Adolf Słabiak, Alfred Schurmann, Jerzy Swianiewicz, Piotr Szorc, Hanna Szymańska-Mysior, Jerzy Waśniewski, Józef Winkowski, Andrzej Wiśniewski, Zdzisława Wrotek, Ryszard Zieliński, Zofia Zjawin-Winkowska. Nie sposób jednak wymienić wszystkich, bioracych udział w oprogramowaniu XYZ; zespół ulegał zmianom, jedni jego członkowie przechodzili do innych zadań, inni włączali się do prac tylko w pewnych fazach projektu. Im wszystkim, także tym nie wymienionym z nazwiska, oprogramowanie XYZ zawdzięczało swój ostateczny kształt.

Wypada wspomnieć o pierwszych zewnętrznych użytkownikach XYZ. Byli nimi młodzi ludzie, obecnie profesorowie Zdzisław Szymański (fizyk) i Andrzej Ziabicki (inżynier fizyko-chemik). Profesor Ziabicki pozostał do dziś entuzjastą komputeryzacji. Obaj oni bez wahania zaakceptowali nową technikę liczenia i

szybko zorientowali się jak komputer może pomóc w rozwiązaniu ich zagadnień. Być może dlatego współpraca z nimi przebiegała pomyślnie, co nie było bynajmniej regułą. Jednym z pierwszych programów użytkowych był program modelujący prace kolejowej "górki rozrządowej" skonstruowany przez Jerzego Swianiewicza. Program ten był o tyle interesujacy, że nie korzystał z wyjścia na kartach lecz podawał wyniki w postaci kropek na oscyloskopie, a jego argumenty były podawane za pośrednictwem kluczy na pulpicie. Był więc ten program prekursorem obecnych programów interakcyjnych. Przypomnijmy, że możliwych kropek było zaledwie 576. Byli później i inni użytkownicy; wiedza o komputerze rozszerzała się szybko, choć nie zawsze zdawano sobie sprawę jak należy wykorzystywać komputer w sposób właściwy. Przykładem typowego błędu było żądanie wyprowadzenia ogromnej ilości liczb, spośród których potem sam użytkownik mozolnie wynajdywał liczbę największą. Do szybkiego rozszerzania się kręgu użytkowników maszyny przyczyniło się SAKO. Możliwość programowania samemu, bez specjalistycznego przygotowania, bez konieczności tłumaczenia problemu nie wprowadzonemu w zagadnienie programiście, była niezmiernie cenna dla licznych nowocześnie myślących użytkowników. Pomocy udzielał im BOP - Biuro Obliczeń i Programów, prowadzone najpierw przez Jerzego Wiśniewskiego, a potem Krzysztofa Moszyńskiego.

Siedząc w 1988 roku nad klawiaturą komputera osobistego i patrząc na ekran monitora, w czasach gdy w kioskach piętrzą się wydawnictwa komputerowe, gdy nastolatki zawzięcie dyskutują o tajnikach systemów operacyjnych, nie chce się wierzyć w sukces oprogramowania takiej maszyny jak XYZ. A jednak niewątpliwie taki sukces osiągnięto. Czynnikiem, który to umożliwił był entuzjazm wszystkich, młodych podówczas, ludzi związanych z maszyną i atrakcyjność nowo tworzącego się zawodu. Gra z maszyną cyfrową, swego rodzaju pojedynek intelektualny z automatem wciągał jak hazard; uprzemysłowienie procesu programowania osłabiło nieco tę atrakcyjność, ale jednak informatyka była i pozostała wielką przygodą

intelektualną niezależnie od tego czy narzędziem jest Cray, VAX, XT, czy stary XYZ.