Mainframe i z/OS, część 2 z 9: Zakład o pięć miliardów: narodziny System/360

Posłuchaj strony

Posłuchaj strony

Zakład o pięć miliardów: narodziny System/360

Około 1960 roku IBM miał problem, który dziś nazwalibyśmy długiem technicznym na poziomie organizacyjnym. Firma sprzedawała równolegle kilka całkowicie niekompatybilnych linii komputerów: **1401** dla małego biznesu, serię **7000** (7070, 7080, 7090 — wzajemnie niezgodne!) dla dużych instalacji, **1620** dla nauki. Program napisany dla jednej nie działał na drugiej. Peryferia nie pasowały. Wsparcie techniczne trzeba było mnożyć przez liczbę linii. Klient, który wyrósł ze swojego 1401, musiał przepisać wszystko od zera — więc równie dobrze mógł się rozejrzeć u konkurencji.

Raport SPREAD

W 1961 roku powołano wewnętrzną grupę zadaniową **SPREAD** (Systems Programming, Research, Engineering And Development) — kilkunastoosobową, z przewodniczącym **Johnem Haanstrą** i wiceprzewodniczącym **Bobem Evansem**; wśród nich **Frederick P. Brooks Jr.** Raport końcowy nosi datę **28 grudnia 1961**.

Warto zauważyć, że to nie była harmonijna narada mędrców. Evans już w maju 1961 rekomendował zabicie projektu „8000” — kolejnej niekompatybilnej linii powstającej w Poughkeepsie. Brooks, który kierował zespołem projektowym tej linii, **początkowo był temu przeciwny** — a potem, poproszony przez Evansa, objął kierownictwo prac nad „rodziną docelową” i to jego entuzjazm przełamał opór reszty. Wiceprezes **T. Vincent Learson** użył grupy SPREAD

jako narzędzia ponadwydziałowego, żeby zneutralizować wojnę między ośrodkami w Endicott i Poughkeepsie — łącznie z oporem samego przewodniczącego grupy.

Rekomendacje raportu: **pięć kompatybilnych procesorów** pokrywających zakres wydajności w stosunku 200:1, **ośmiobitowy bajt** jako jednostka podstawowa, pamięć mikroprogramu tylko do odczytu, **jeden standardowy interfejs dla wszystkich urządzeń peryferyjnych**, arytmetyka zmiennoprzecinkowa. Jedna architektura dla nauki i dla biznesu.

Pięć miliardów dolarów

We wrześniu 1966 magazyn *Fortune* opublikował dwuczęściowy artykuł T.A. Wise'a pod tytułem, który przeszedł do historii: **„IBM's $5,000,000,000 Gamble”**. Liczby, które warto zestawić:

- koszt programu: około **5 miliardów dolarów** (rzędu 40 miliardów dzisiejszych);
- roczne przychody IBM w 1964: **3,2 miliarda** — czyli projekt kosztował więcej niż całoroczna sprzedaż firmy;
- Projekt Manhattan kosztował około 2 miliardów ówczesnych dolarów — porównanie nominalne, ale często i nie bez powodu przywoływane;
- zatrudniono **ponad 70 tysięcy nowych pracowników**; stan osobowy IBM urósł ze 127 tysięcy (1962) do 265 tysięcy (1971);
- zbudowano nowe zakłady, w tym East Fishkill, który w pewnym momencie produkował więcej elementów półprzewodnikowych niż wszyscy pozostali producenci świata razem wzięci.

To jest dosłowne znaczenie zwrotu *„you bet your company”*. Gdyby System/360 się nie przyjął, IBM nie miałby do czego wrócić — właśnie wygaszał wszystkie dotychczasowe linie produktowe.

7 kwietnia 1964

Ogłoszenie przeprowadzono jednocześnie na konferencjach prasowych w kilkudziesięciu miastach USA i kilkunastu krajach (najczęściej podaje się 63 miasta i 14 krajów; liczba jest powszechnie powtarzana, choć trudna do niezależnego potwierdzenia). Zaprezentowano **sześć modeli** — 30, 40, 50, 60, 62 i 70 — oraz kilkadziesiąt urządzeń peryferyjnych. Ceny: czynsz miesięczny od 2700 do 115 000 dolarów, zakup od 133 000 do 5,5 miliona.

> **Ciekawostka — drewniane makiety.** Część urządzeń peryferyjnych pokazywanych dziennikarzom 7 kwietnia 1964 roku to były **atrapy z drewna**. Projekt wciąż trwał, a IBM ogłaszał produkt, którego jeszcze nie potrafił zbudować. To zresztą stanie się później zarzutem sądowym w innej sprawie (patrz rozdział 18): ogłaszanie „maszyn papierowych”, żeby zniechęcić klientów do zamówień u konkurencji.

Reakcja rynku: **ponad tysiąc zamówień w pierwszych czterech tygodniach**, a po dwóch latach tempo sięgnęło tysiąca zamówień miesięcznie. Nazwa „360” odsyłała do pełnego kąta —

wszystkich kierunków kompasu, wszystkich zastosowań, naukowych i komercyjnych naraz.

Realizacja była trudniejsza niż ogłoszenie. Już w październiku 1964 IBM przyznał się do poważnych opóźnień. W 1965 roku **25% modułów SLT** (Solid Logic Technology — hybrydowa technologia układów, którą IBM musiał wymyślić na potrzeby projektu) odrzucała kontrola jakości. Oprogramowanie było w jeszcze gorszym stanie — o czym cały rozdział 7. Do końca 1966 dostarczono jednak 7700 systemów, a do 1970 zainstalowana baza sprzętu IBM była warta 24 miliardy dolarów wobec 9 miliardów u całej konkurencji łącznie.

Architekci i ich spory

Trzej ludzie odpowiadają za kształt tej architektury: **Gene Amdahl** (główny architekt), **Frederick P. Brooks Jr.** (kierownik projektu, potem szef OS/360) i **Gerrit Blaauw**. Ich wspólny artykuł „Architecture of the IBM System/360” z *IBM Journal of Research and Development* (1964) jest do dziś czytany na kursach architektury komputerów.

Dwie ich decyzje ukształtowały całą późniejszą informatykę — jedna genialnie, druga katastrofalnie.

Ośmiobitowy bajt. To Blaauw przeforsował ośmiobitową jednostkę adresowalną wobec ówczesnego standardu sześciu bitów i presji kosztowej, by zostało przy czterech lub sześciu. Ponieważ System/360 zdominował rynek, ośmiobitowy bajt stał się standardem świata. *Każdy* plik, protokół i procesor, z którym dziś pracujesz, liczy w bajtach po osiem bitów, bo w 1963 roku jeden inżynier w Poughkeepsie wygrał spór.

Adresowanie 24-bitowe. Adres miał 24 bity, czyli 16 megabajtów przestrzeni. W 1964 roku, gdy typowa maszyna miała 32 kilobajty pamięci, było to *pięćset razy więcej, niż ktokolwiek potrzebował* — decyzja wyglądała na rozrzutną. Osiemnaście lat później ta sama decyzja wymusiła najbardziej bolesną operację w historii tej architektury (rozdział 20) i zostawiła w żargonie mainframe'owym terminy *„above the line”* i *„below the line”*, żywe do dziś.

> **Ciekawostka — bit ASCII, który nigdy nie zadziałał.** System/360 wprowadził własne kodowanie znaków **EBCDIC**, wywodzące się wprost ze stref i cyfr karty dziurkowanej — mimo że ASCII właśnie wystandaryzowano (1963). Architekci przewidzieli jednak wyjście awaryjne: w Program Status Word umieszczono **bit trybu ASCII** (słynny „P-bit”), który miał przełączać maszynę na nowy standard. Jak napisał później **Bob Bemer**, jeden z twórców ASCII: *„the 360 series software would not, and could not, run in ASCII mode, even when the infamous 'P-bit' was switched”* — całe oprogramowanie systemowe było napisane pod EBCDIC, więc bit istniał, ale nie robił nic. Bemer podsumował skutki decyzji IBM krótko: *„look at the countless billions that has cost the world”*. EBCDIC żyje do dziś i ma zabawną właściwość: **alfabet w nim nie jest ciągły** — litery A–I, J–R i S–Z leżą w trzech osobnych zakresach z dziurami (bo odpowiadają trzem kombinacjom stref na karcie). Naiwna pętla „od A do Z” napotyka w EBCDIC 41 pozycji kodowych zamiast 26.

Sztuczka, która sprzedała maszynę: emulacja w mikrokodzie

IBM 1401 w Computer History Museum

IBM 1401 — najpopularniejszy komputer świata przed 1964 rokiem. System/360 potrafił udawać go w mikrokodzie, co pozwoliło klientom kupić nową maszynę bez przepisywania starego oprogramowania. Źródło: Wikimedia Commons.

Najważniejszy argument handlowy System/360 nie był architektoniczny, tylko migracyjny — i był genialny w swojej prostocie. Prawie wszystkie modele rodziny implementowały zestaw instrukcji **w mikrokodzie** zapisanym w pamięci sterującej. Inżynierowie z Endicott wpadli na to, żeby dopisać do tej pamięci **cały zestaw instrukcji IBM 1401**.

Efekt: klient wstawiał nowy System/360, przełączał go w „tryb 1401” i uruchamiał swoje stare, nieskompilowane na nowo programy — **nawet czterokrotnie szybciej niż na oryginalnym 1401**. Model 30 emulował 1401, 1410 i 7010; Model 65 emulował całą serię 7000. Nazwano to „spełnieniem marzenia sprzedawcy”. Problem, który raport SPREAD zdiagnozował jako największą bolączkę oferty IBM — niemożność migracji — rozwiązano dokładając linijki mikrokodu.

Cena sukcesu: Amdahl odchodzi

Warto odnotować dwie porażki na marginesie triumfu. **Model 67** — pierwszy system IBM ze sprzętową translacją adresów, czyli pamięcią wirtualną — miał obsługiwać system podziału czasu **TSS/360**, który anulowano w 1968, wskrzeszono w 1969 i ostatecznie zamknięto w 1971. Paradoksalnie to właśnie na osieroconym Modelu 67 rozkwitły systemy, które przetrwały: CP/CMS (o czym rozdział 8), MTS i MUSIC.

Gene Amdahl po System/360 poprowadził laboratorium ACS w Menlo Park, projektując architekturę kolejnej generacji. Gdy jego koncepcje odrzucono, **we wrześniu 1970 roku odszedł z IBM** — żeby założyć firmę, która będzie budować maszyny zgodne z IBM i konkurować z nim ceną. O tym w rozdziale 17.

Architektura od środka

Ten rozdział jest dla tych, którzy chcą wiedzieć, *jak to jest zbudowane*. Architektura System/360 jest fascynująca, bo powstała przed większością konwencji, które dziś uważamy za oczywiste — i część jej rozwiązań jest po prostu inna, a nie gorsza.

Rejestry i PSW

System/360 dał programiście **16 rejestrów ogólnego przeznaczenia** po 32 bity (numerowane 0–15) oraz **4 rejestry zmiennoprzecinkowe** po 64 bity — te ostatnie numerowane nietypowo: 0, 2, 4, 6, bo format instrukcji rezerwował bit na rozróżnianie par.

Sercem procesora jest **PSW** — *Program Status Word*, 64-bitowe słowo zawierające *cały* stan wykonania:

W z/Architecture PSW rozrosło się do 128 bitów, z adresem instrukcji na pełnych 64 bitach i dodatkowymi bitami wybierającymi tryb adresowania (24-, 31- lub 64-bitowy). Ale zasada została: **jedno słowo opisuje wszystko**, a przełączenie kontekstu to w istocie podmiana PSW.

Formaty instrukcji: RR, RX, RS, SI, SS

Instrukcje mają **zmienną długość: 2, 4 albo 6 bajtów** — na tle ówczesnych architektur o stałej długości słowa to była nowość. Pięć podstawowych formatów:

- **RR** (2 bajty) — rejestr–rejestr: `AR R1,R2`, a także `SVC n`.
- **RX** (4 bajty) — rejestr–pamięć z indeksowaniem: `L R1,D2(X2,B2)`.
- **RS** (4 bajty) — dwa rejestry i adres, np. `LM` (load multiple).
- **SI** (4 bajty) — operand pamięciowy plus wartość natychmiastowa: `MVI`, `CLI`, `TM`.
- **SS** (6 bajtów) — *pamięć do pamięci* ze wskazaniem długości: `MVC`, `CLC`, `PACK`, `ED`, oraz cała arytmetyka dziesiętna.

Baza i przesunięcie, czyli dlaczego programiści mówią o USING

Tu jest cecha, która zaskakuje każdego, kto uczył się asemblera na x86: w System/360 **nie ma adresowania bezwzględnego**. Każdy adres pamięci w instrukcji to para `D(B)` albo `D(X,B)`: *rejestr bazowy* plus *12-bitowe przesunięcie bez znaku*. Dwanaście bitów to zakres 0–4095 — stąd magiczna liczba **4096 bajtów zasięgu z jednego rejestru bazowego**.

Konsekwencja praktyczna: każdy program w asemblerze musi jawnie zadeklarować, który rejestr pełni rolę bazy, i faktycznie go załadować:

Jeśli sekcja kodu albo danych przekroczy 4096 bajtów od bazy, trzeba drugiego rejestru bazowego — i to jest źródło klasycznego komunikatu `ADDRESSABILITY ERROR`, na którym poległo kilka pokoleń początkujących. Dopiero rozszerzenie „long displacement” w S/390 dało instrukcje z 20-bitowym przesunięciem ze znakiem (te z sufiksem Y, np. `LY`), łagodząc problem — ale nie likwidując samego pojęcia rejestru bazowego, które siedzi w tej architekturze od 1964 roku.

Klucze ochrony pamięci: mechanizm, którego pecet nie ma

Oto rozwiązanie, które nie ma odpowiednika w świecie x86 i które warto zrozumieć, bo tłumaczy sporo z kultury bezpieczeństwa mainframe'a.

Każdy blok pamięci fizycznej ma przypisany **czterobitowy klucz ochrony** (16 możliwych wartości). PSW niesie **klucz bieżącego procesu**. Reguła zapisu: wolno pisać, gdy klucz PSW wynosi 0 (klucz nadrzędny), gdy klucz PSW równa się kluczowi bloku, albo gdy blok ma klucz 9 (publiczny). Odczyt jest bardziej permisywny, chyba że blok ma ustawiony bit ochrony przed odczytem.

W z/OS przyjęto, że **klucze 0–7 należą do systemu, a 8–15 do aplikacji** (domyślny klucz programu użytkownika to 8). Przejście na klucz uprzywilejowany wymaga jednoczesnego bycia w stanie nadzorcy — stąd w świecie z/OS trzy odrębne, notorycznie mylone pojęcia: **APF-authorized** (moduł leży w autoryzowanej bibliotece), **supervisor state** (bit 15 PSW) i **key 0**. To nie są synonimy i nie wynikają z siebie automatycznie.

Efekt architektoniczny: mainframe ma **znacznie drobniejszą granulację izolacji niż podział user/kernel**. CICS, Db2 i sam system mogą działać „w jądrze”, a mimo to być od siebie sprzętowo oddzielone kluczami. To jest ten sam duch, co cztery pierścienie ochrony — tylko zrealizowany inaczej.

Przerwania i wejście do jądra

Jedyna legalna droga z trybu aplikacji do jądra prowadzi przez instrukcję `SVC n` (Supervisor Call) — dokładny odpowiednik `syscall`. Mechanizm: stare PSW zapisywane pod stały adres w niskiej pamięci, nowe PSW ładowane z sąsiedniego adresu. Dlatego pierwsze kilkaset bajtów pamięci mainframe'a ma twardo zakodowane znaczenie sprzętowe (obszar zwany PSA).

Architektura zna **sześć klas przerwań**, i ten podział przetrwał bez zmian od 1964 roku:

1. **SVC** — żądanie usługi systemowej;
2. **Program** — błąd programu: zły kod operacji, naruszenie ochrony, błąd strony, przepełnienie;
3. **External** — zegar, przycisk RESTART na konsoli, sygnał z innego procesora;
4. **I/O** — kanał zgłasza zakończenie operacji;
5. **Machine check** — awaria sprzętu wykryta przez logikę RAS;
6. **Restart** — interwencja operatora lub innego procesora.

Arytmetyka dziesiętna, czyli dlaczego bank liczy inaczej

Tu dochodzimy do cechy, która tłumaczy, dlaczego ta architektura przetrwała w finansach. Bank nie może sobie pozwolić, żeby 0,1 + 0,2 dało 0,30000000000000004. System/360 ma więc **sprzętową arytmetykę dziesiętną**:

- **zoned decimal** — po jednej cyfrze na bajt (strefa + cyfra), ostatni bajt niesie znak; to naturalny format tekstowy EBCDIC;
- **packed decimal** — dwie cyfry BCD na bajt, ostatni półbajt to znak; instrukcje `PACK`/`UNPK` konwertują, a `AP`, `SP`, `MP`, `DP`, `CP` liczą wprost na tym formacie.

I perła: instrukcje **`ED` (Edit) i `EDMK` (Edit and Mark)**. Biorą liczbę w packed decimal i wzorzec formatujący, po czym *jedną instrukcją* tłumią zera wiodące, wstawiają separatory tysięcy i przecinek dziesiętny, obsługują znak i wypełniacz. `EDMK` dodatkowo zwraca w rejestrze adres pierwszej znaczącej cyfry, żeby dało się wstawić symbol waluty dokładnie przed nią. Klasyczny wynik `***1,234.56` z gwiazdkami zabezpieczającymi czek — jedna instrukcja procesora. Formatowanie raportów finansowych zaszyte w krzemie, w 1964 roku.

W praktyce wygląda to tak — fragment w asemblerze doliczający 23% podatku i formatujący kwotę brutto do wydruku:

Cztery instrukcje na pełne wyliczenie z zaokrągleniem bankowym i jedna na sformatowanie wyniku. `SRP` (Shift and Round Packed) przesuwa przecinek *z zaokrągleniem* — też jedną instrukcją. To jest procesor zaprojektowany przez ludzi, którzy rozumieli, że komputer w banku spędza życie na mnożeniu kwot przez stawki.

> **Ciekawostka — zmiennoprzecinkowa, która chwieje precyzją.** System/360 użył formatu zmiennoprzecinkowego o **podstawie 16**, a nie 2 — bo przy normalizacji do cyfry szesnastkowej rzadziej trzeba przesuwać mantysę, co było szybsze i tańsze w tanich modelach. Cena: *wobbling precision*. Mantysa może mieć do trzech wiodących bitów zerowych, więc efektywna precyzja pojedyncza waha się między 21 a 24 bitami zależnie od wartości liczby. Wczesne implementacje podwójnej precyzji nie miały nawet cyfry strażniczej przy zaokrąglaniu — IBM musiał to naprawiać już po dostawach do klientów, a wiele operacji do dziś ucina zamiast zaokrąglać, „jak prosty kalkulator kieszonkowy”. Binarną arytmetykę zgodną z IEEE 754 mainframe dostał dopiero w **1998** (S/390 G5), a sprzętową **dziesiętną zmiennoprzecinkową IEEE** — w 2007–2008 (z9 GA2, potem pełny krzem w z10). To rzadkość: dziś rdzeń IBM Z ma *trzy* formaty zmiennoprzecinkowe naraz (HFP, BFP, DFP), bo zgodności wstecznej nigdy się nie usuwa.

Kanały i dyski: osobny komputer do wejścia-wyjścia

Jeśli jedna idea miałaby zdefiniować mainframe'a, byłby to **kanał**. To jest cecha, która najlepiej tłumaczy, dlaczego ta maszyna do dziś robi rzeczy, których serwer x86 nie robi.

Kanał jako procesor

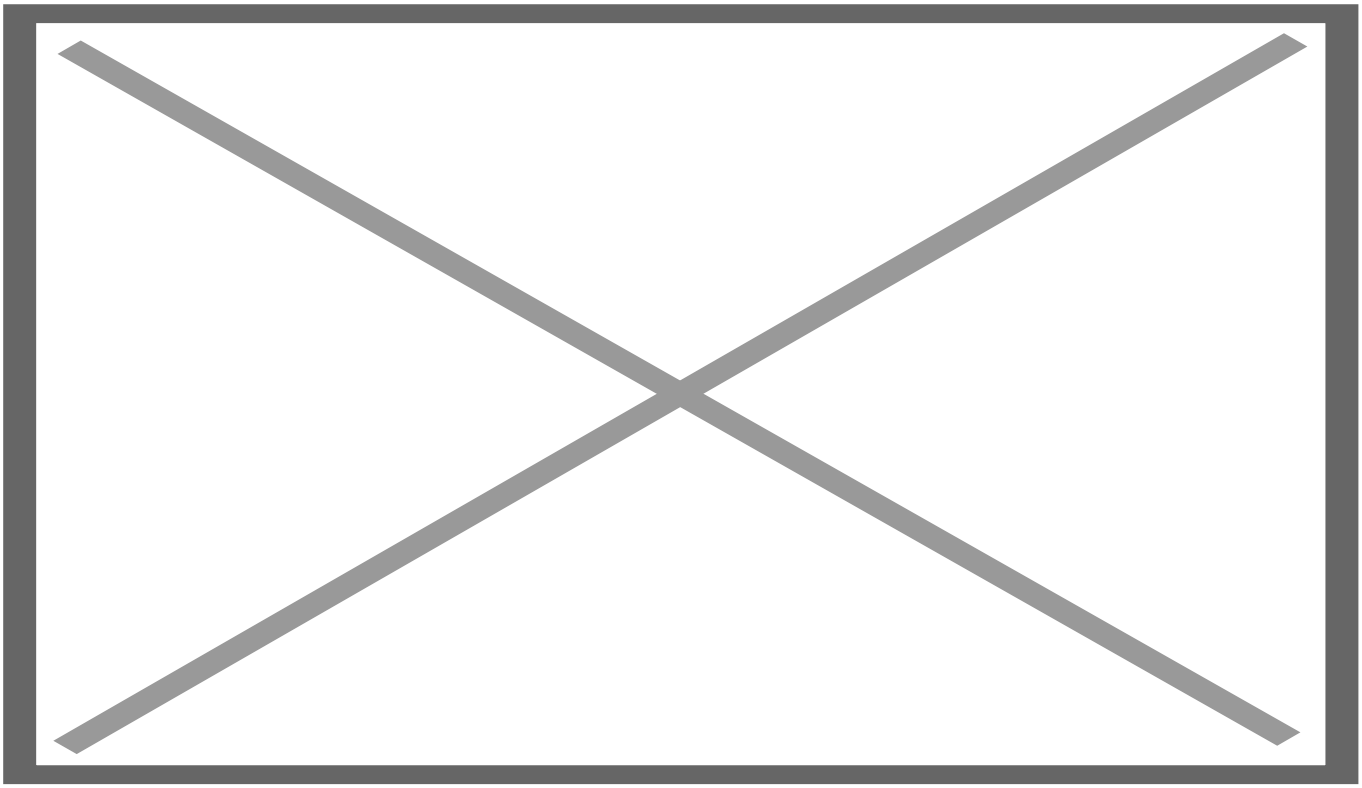
Napędy taśmowe IBM 729

Napędy taśmowe IBM 729 w ośrodku NASA. Każdy z nich wisiał na kanale — osobnym procesorze wejścia-wyjścia, który wykonywał własny program (CCW) bez udziału jednostki centralnej. Źródło: Wikimedia Commons.

Kanał nie jest magistralą. Kanał to **osobny, programowalny procesor wejścia-wyjścia**, który wykonuje własny program złożony z **CCW** (Channel Command Word) — instrukcji typu READ, WRITE, SEEK, TIC (skok wewnątrz programu kanałowego).

Procesor główny wykonuje **jedną uprzywilejowaną instrukcję** — historycznie `SIO` (Start I/O), dziś `SSCH` (Start Subchannel) — przekazując wskaźnik do bloku opisującego program kanałowy. I na tym kończy się jego udział. Kanał autonomicznie realizuje *cały* transfer: wybiera ścieżkę, obsługuje przeszukiwanie dysku, ponawia próby po błędach, przetwarza łańcuch poleceń — a na koniec zgłasza **jedno przerwanie**.

Porównanie dla administratora Linuksa: nawet z `io_uring` główny procesor musi zaprogramować kontroler przez sterownik, obsłużyć przerwanie (choćby krótkie) i zarządzić obsługą błędów w kodzie jądra wykonywanym na tym samym CPU. Na mainframie *cała* ta logika siedzi w oddzielnym krzemie. To nie jest DMA — to jest **offloading pełnej ścieżki I/O**, koncepcyjnie bliższy dzisiejszym SmartNIC-om i DPU, tylko wdrożony dla *każdej* operacji w systemie od 1964 roku.



Animacja 1. *Kanał jako procesor: jedna instrukcja na start, jedno przerwanie na końcu. Wszystko pomiędzy — wybór ścieżki, szukanie rekordu po kluczu, ponowienie po błędzie, a nawet obsługa innego urządzenia w czasie ruchu głowicy — dzieje się w kanale, gdy CPU liczy dalej (łańcuch CCW i skala czasu przykładowe).*

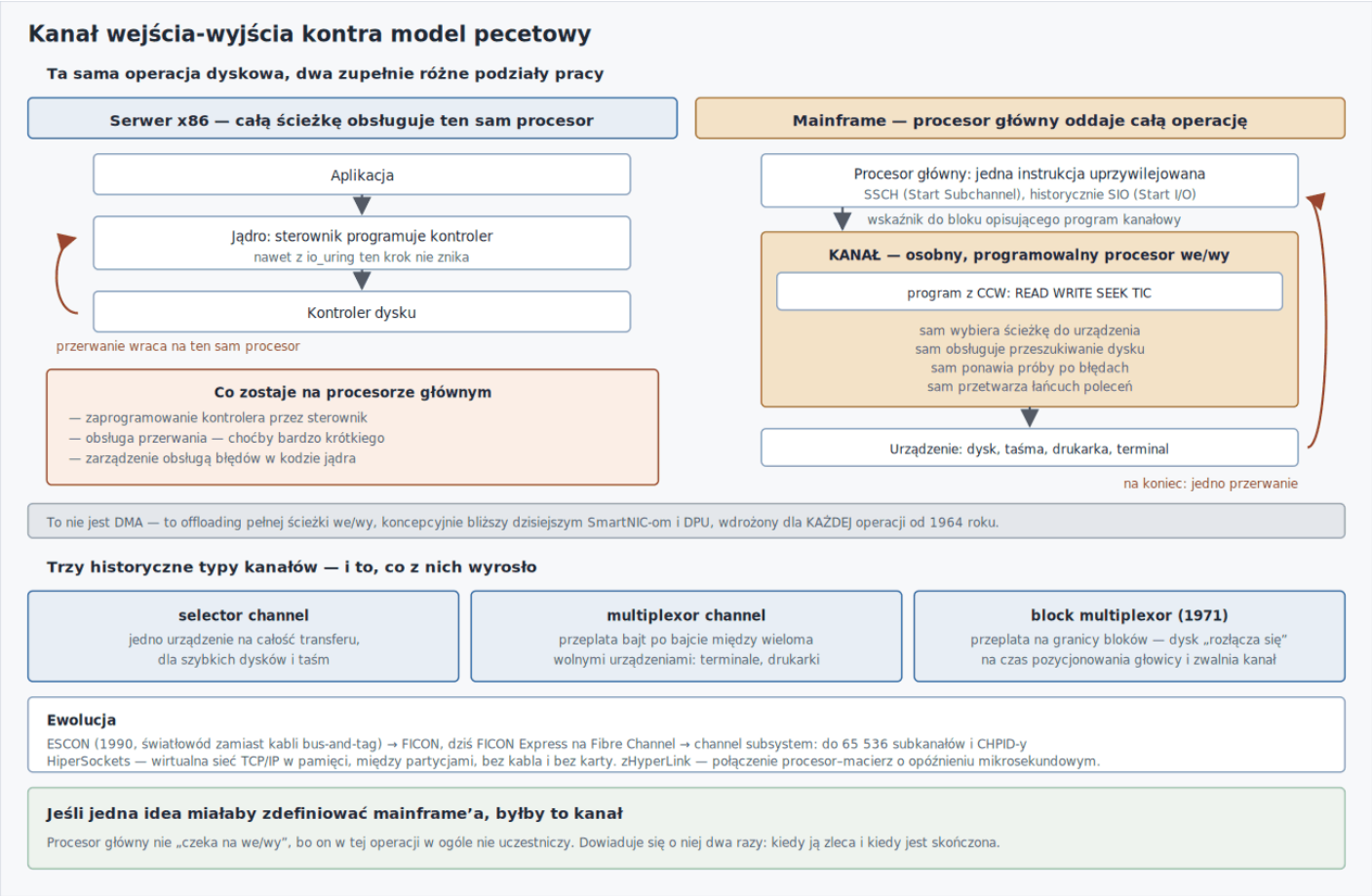
Historyczne typy kanałów mówią sporo o myśleniu projektantów:

- **selector channel** — jedno urządzenie na całość transferu, dla szybkich dysków i taśm;
- **multiplexor channel** — przeplata bajt po bajcie między wieloma wolnymi urządzeniami (terminale, drukarki);
- **block multiplexor channel** (1971) — przeplata na granicy bloków, dzięki czemu dysk może się „rozłączyć” od kanału na czas mechanicznego pozycjonowania głowicy, zwalniając go dla innych urządzeń. Jeden z tych pomysłów, które wyglądają na oczywiste dopiero, gdy ktoś je wymyśli.

Okablowanie mainframe'a w Living Computer Museum w Seattle (2013)

Okablowanie mainframe'a w Living Computer Museum w Seattle (2013). Źródło: Flickr, CC BY-NC-ND 2.0, autor: Matthew Ratzloff.

Ewolucja: **ESCON** (1990, pierwsze połączenia światłowodowe zamiast grubych kabli bus-and-tag), **FICON** (od końca lat 90., dziś FICON Express na Fibre Channel), **channel subsystem** z subkanałami (do 65 536 na maszynę) i CHPID-ami, **HiperSockets** (wirtualna sieć TCP/IP realizowana w pamięci między partycjami, bez kabla i bez karty) oraz **zHyperLink** (bezpośrednie połączenie procesor–macierz o opóźnieniu mikrosekundowym, z pominięciem stosu FICON dla operacji krytycznych czasowo, jak zapis logu transakcyjnego).



Rysunek 1. *Różnica nie polega na szybkości transferu, tylko na tym, kto wykonuje program obsługujący transfer: na mainframie robi to osobny krzem, a jednostka centralna dowiaduje się o wszystkim jednym przerwaniem na końcu.*

Dysk, który jest inny: CKD

Moduł dyskowy IBM 3380

Moduł dyskowy IBM 3380 z połowy lat 80. Format CKD (Count-Key-Data) pozwalał kanałowi samodzielnie szukać rekordu po kluczu — dysk brał na siebie część pracy bazy danych. Źródło: Wikimedia Commons.

Dysk mainframe'owy nie ma sektorów stałej długości. Ma format **CKD** — *Count-Key-Data* — w którym **każdy rekord opisuje sam siebie**:

- **Count** — 8 bajtów: identyfikator rekordu w postaci **CCHHR** (cylinder, głowica, numer rekordu) plus długości pozostałych pól;
- **Key** — do 255 bajtów klucza, po którym *kanał* potrafi wyszukiwać rekord bez angażowania procesora;
- **Data** — dane o dowolnej długości.

Przeciwieństwem jest **FBA** (Fixed Block Architecture) — jednolite sektory, jak w każdym dysku PC. CKD pozwalał dopasować rekord do potrzeb aplikacji i przerzucić wyszukiwanie po kluczu na

kontroler; FBA jest prostsze i tańsze. Geometrie kolejnych generacji — 3330, 3350, 3380, **3390** — weszły do żargonu na stałe, a wolumin identyfikuje sześcioznakowy **VOLSER**.

> **Ciekawostka — dyski, których nie ma.** Fizyczne dyski CKD praktycznie nie istnieją od dawna. Współczesna macierz (np. IBM DS8000) trzyma dane na zwykłych SSD w architekturze blokowej, a **geometrię 3390 syntetyzuje w pamięci kontrolera** i mapuje na bloki stałej długości. Czyli: z/OS w 2026 roku adresuje dane przez numer cylindra i głowicy nieistniejącego talerza magnetycznego z lat 80., a macierz udaje, że to prawda. Do tego dochodzą **Extended Address Volumes**, które rozszerzyły adresowanie cylindra daleko poza pierwotny limit, żeby udawany dysk 3390 mógł mieć wiele terabajtów. Warstwa zgodności zbudowana na warstwie zgodności — i wszystko działa.

Revision #2
Created 2026-11-03 00:54:54 UTC by Tomasz Konieczny
Updated 2026-11-03 02:49:47 UTC by lektor