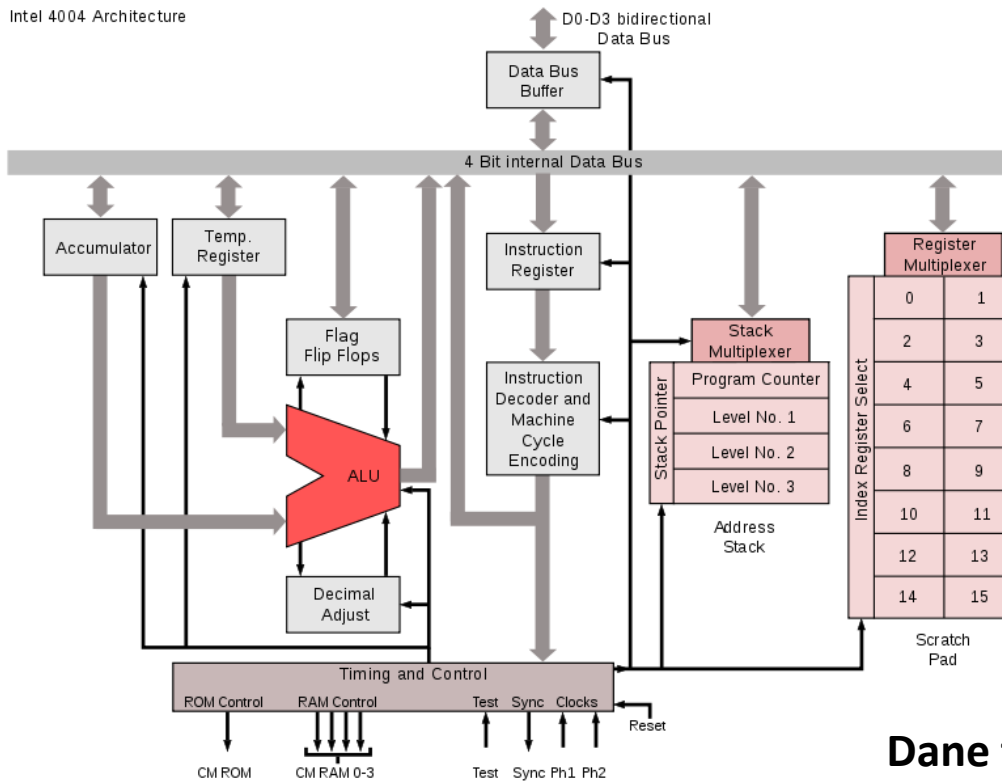


## Intel 4004

- wprowadzony 15 listopada 1971
- zegar: 740 kHz
- moc obliczeniowa: 0,09 MIPS
- szyna danych: 4-bitowa
- liczba tranzystorów 2300, 10 mikronów
- pamięć adresowalna 640 bajtów
- pamięć programu 4 kilobajty
- pierwszy mikroprocesor na świecie
- używany w kalkulatorach Busicom

# Intel 4004

Intel 4004 Architecture



## Dane techniczne:

- Osobna pamięć dla programu i danych (tzw. "architektura harwardzka").
- 46 instrukcji.
- 16 czterobitowych rejestrów
- 3-poziomowy stos

# Intel 8080

Jest on uniwersalną jednostką centralną złożoną z jednostki arytmetyczno-logicznej, rejestrów roboczych i układu sterowania.

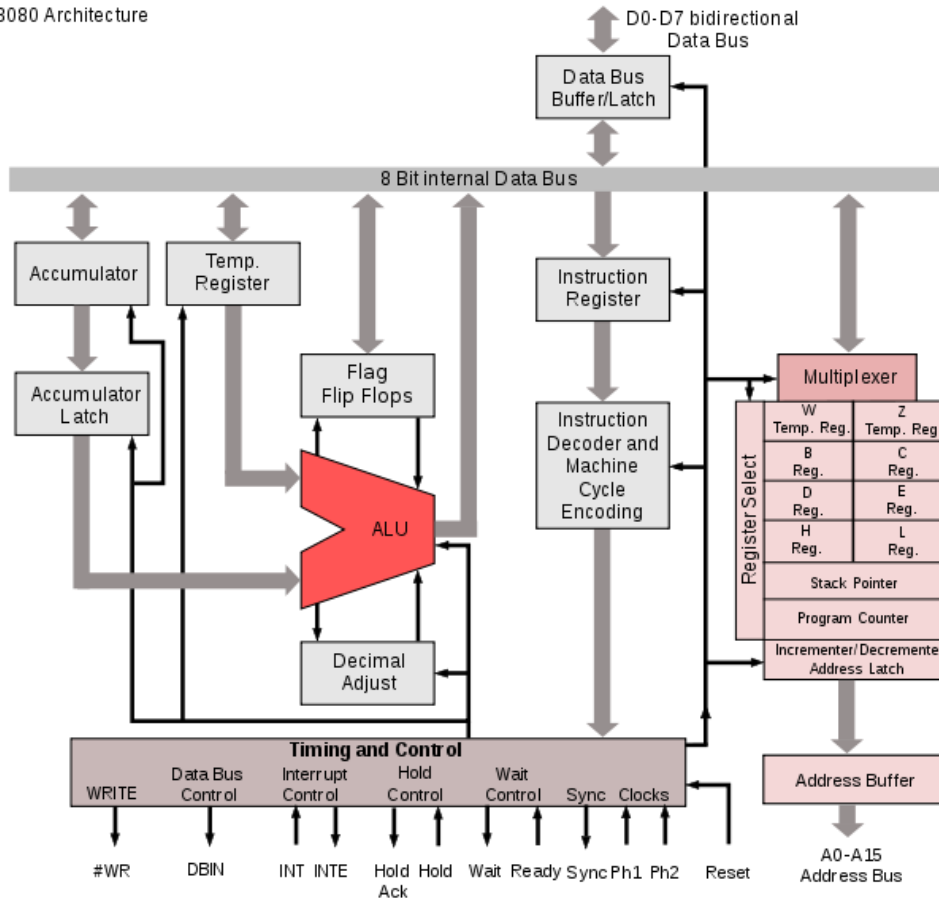
- wyprodukowany w kwietniu 1974
- zegar 2 MHz
- moc obliczeniowa 0,64 MIPS
- szyna danych 8-bitowa
- liczba tranzystorów 6000, 6 mikronów
- pamięć jest adresowana 16-bitową szyną adresową
- pamięć adresowalna 64 kB

# Intel 8080

- słowo 8-bitowe
- 72 instrukcje
- arytmetyka dwójkowa i dziesiętna kodowana dwójkowo (BCD)
- 8 rejestrów programowych dostępnych dla programisty
- cykl pracy  $2\mu\text{s}$ , wymuszany przez 2-fazowy zegar zewnętrzny
- częstotliwość zegara 2-3 MHz (podstawowy cykl rozkazowy – 4 takty)
- 3 napięcia zasilające: +5V, +12V, -5V
- ubogi zestaw trybów adresowania
- konieczność stosowania dodatkowych układów: zegar i sterownik systemu

# Intel 8080

Intel 8080 Architecture

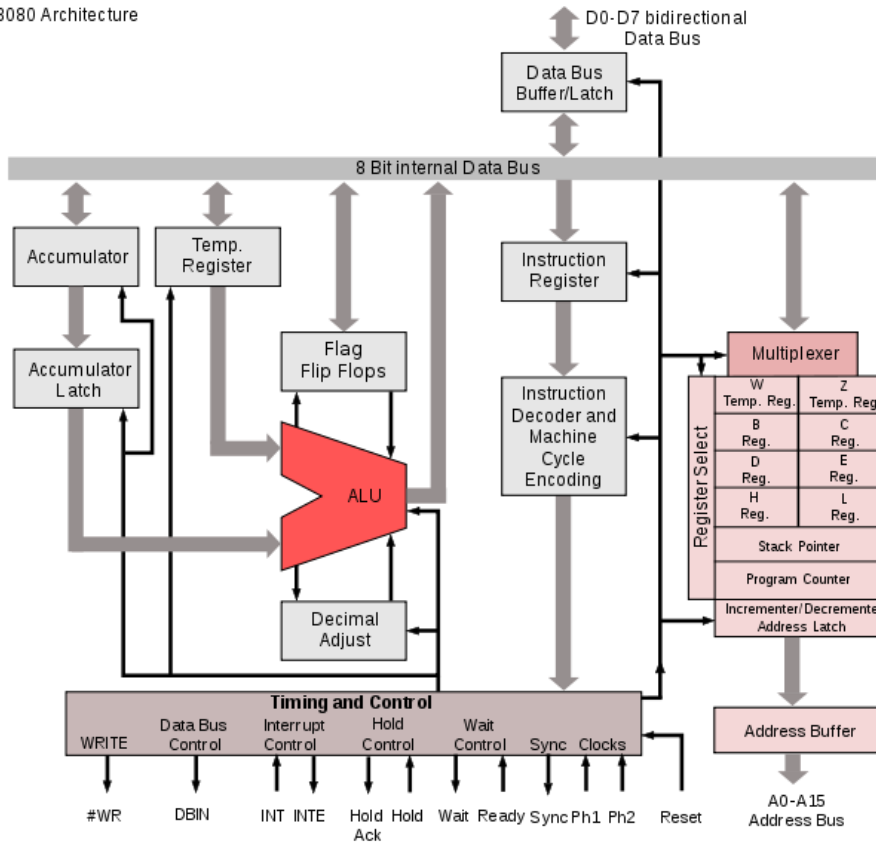


W strukturze mikroprocesora można wyróżnić cztery bloki funkcjonalne:

- blok rejestrów wraz z układem wybierającym
- jednostkę arytmetyczno-logiczną
- układ sterowania z rejestrem rozkazów
- dwukierunkowy, trójstanowy bufor szyny danych

# Intel 8080

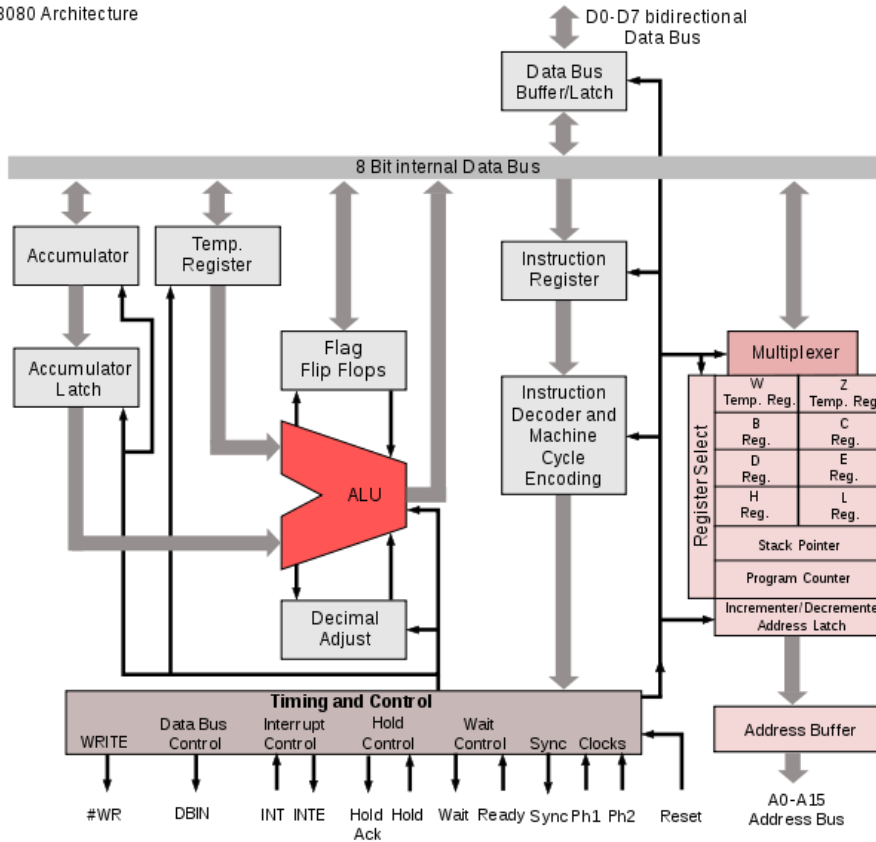
Intel 8080 Architecture



- blok rejestrów z układem wybierającym jest statyczną pamięcią z dostępem swobodnym (RAM), zorganizowaną w ten sposób, że tworzy następujące rejestry:
- licznik rozkazów (PC) – 16-bitowy
- wskaźnik stosu (SP) – 16-bitowy
- sześć ośmiobitowych rejestrów uniwersalnych: B, C, D, E, H, L, które można łączyć w pary
- rejestr pomocniczy składający się z dwóch ośmiobitowych rejestrów W oraz Z nie jest dostępny dla programisty i wykorzystywany jest do operacji wewnętrznych.

# Intel 8080

Intel 8080 Architecture



- jednostka arytmetyczno-logiczna może wykonywać działania w arytmetyce dwójkowej, operując liczbami ośmiobitowymi,
- do realizacji arytmetyki dziesiętnej wprowadzono rozkazy korekcji po dodawaniu i odejmowaniu.
- z jednostką ALU związany jest rejestr znaczników, którego zawartość tworzy warunki pracy przy wykonywaniu operacji arytmetycznych, logicznych i instrukcji warunkowych. Poszczególne bity tego rejestru służą do określenia:

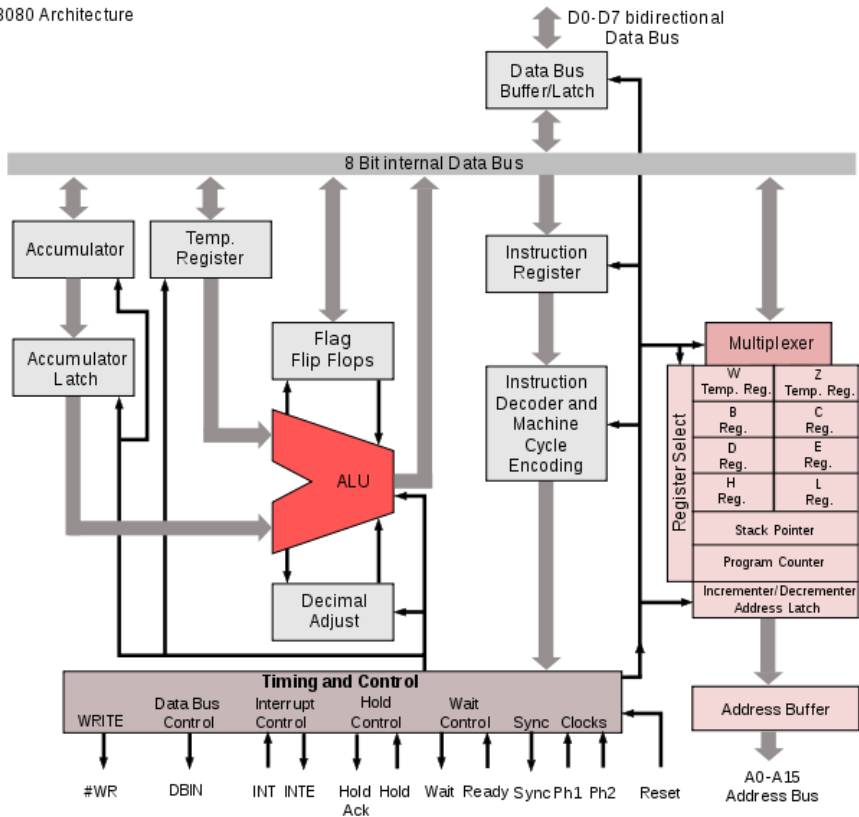
zera na wyjściu sumatora  
przeniesienia

znaku parzystości

przeniesienia półówkowego

# Intel 8080

Intel 8080 Architecture



Bufor szyny danych jest ośmiobitowym, dwukierunkowym rejestrem służącym do odizolowania wewnętrznej szyny danych od końcówek  $D_0 \dots D_7$  stanowiących zewnętrzną szynę danych.

Lista rozkazów 8080 obejmuje: przesłania między rejestrami procesora a pamięcią zaadresowaną w jednym z trzech trybów (natychmiastowy, bezpośredni, zawartością pary rejestrów)



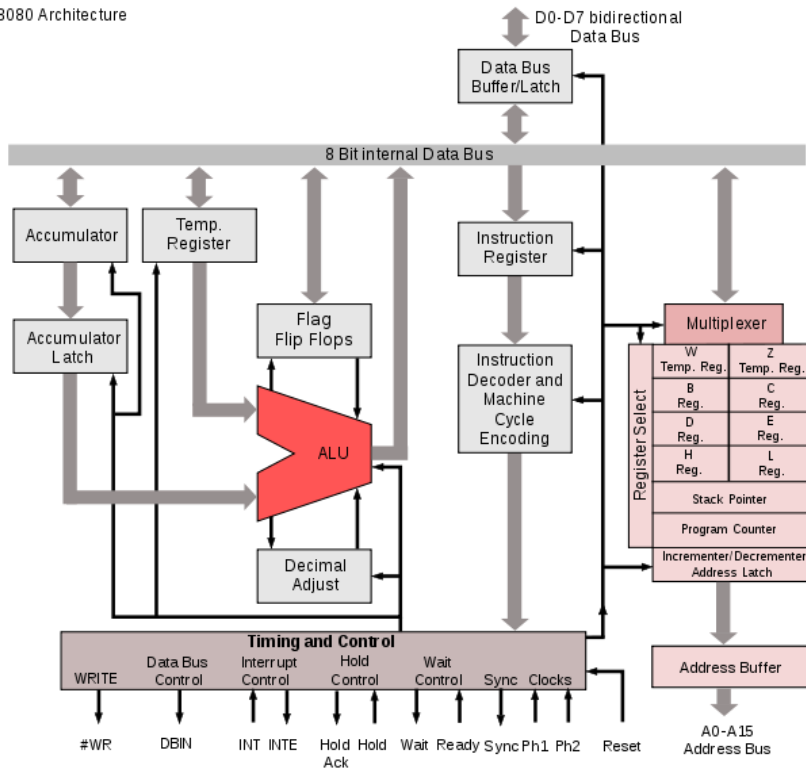
# Intel 8080

Lista rozkazów 8080 obejmuje:

- operacje arytmetyczne dodawania, odejmowania, porównywania wykonywane na ośmiobitowych liczbach dwójkowych, całkowitych bez znaku lub ze znakiem w kodzie U2, jednym z argumentów operacji jest zawartość **akumulatora**, drugim – zawartość rejestru roboczego, argument bezpośredni lub zawartość komórki pamięci zaadresowanej parą H-L
- dodawanie i odejmowanie wielokrotnej precyzji
- operacje logiczne na słowach ośmiobitowych
- rozkazy skoku, przy czym skok warunkowy jest skokiem bezwzględnym, a warunkiem skoku jest stan wybranego znacznika w rejestrze stanu
- rozkazy wejścia/wyjścia realizujące przesłania między akumulatorem a układem sprzęgającym, którego numer jest podany w rozkazie

# Intel 8080

Intel 8080 Architecture



## Układ sterowania z rejestrem rozkazów :

- Rejestr rozkazów jest rejestrem ośmiobitowym, do którego na początku każdego cyklu rozkazowego jest przesyłany pierwszy bajt instrukcji (zawierający kod instrukcji) pobrany z zewnętrznej pamięci przez szynę danych  $D_0 \dots D_7$  i wewnętrzną szynę danych.
- Zawartość rejestru rozkazowego jest dostępna dla dekodera instrukcji, który generuje, wraz z układem sterującym sygnały sterujące dla wszystkich układów mikroprocesora.
- Do układów są doprowadzone końcówki  $\phi 1$ ,  $\phi 2$ , READY, INT, RESET i HOLD, przez które wprowadza się dodatkowe informacje sterujące.
- Z układu sterowania jest wyprowadzonych na zewnątrz sześć wyjść identyfikujących stany i cykle mikroprocesora dla układów zewnętrznych. Są to INTE, HLDA, DBIN, SYNC, WR i WAIT.

# Architektura procesorów CISC kontra RISC

Wśród aktualnie produkowanych procesorów obserwujemy dwa typy :

- **procesory typu CISC ( Compound Instruction Set Computer - komputery o złożonej liście rozkazów)**
- **procesory typu RISC ( Reduced Instruction Set Computer - komputery o zredukowanej liście rozkazów).**

# CISC

Procesory typu CISC mają następujące cechy architekuralne:

- Rozbudowana lista rozkazów zawierająca od 100 do 300 rozkazów wewnętrznych.
- Wiele rozkazów wewnętrznych ma skomplikowaną treść operacyjną, realizują one w jednym rozkazie skomplikowane operacje łączące dostępy do pamięci operacyjnej z przetwarzaniem danych.
- Duża liczba trybów adresowania dostępna w rozkazach wewnętrznych, od 5 do 20.
- Mała liczba rejestrów roboczych w procesorze, od kilku do kilkunastu.
- Formaty rozkazów wewnętrznych zróżnicowane pod względem: podziału na pola, długości słowa rozkazowego i liczby argumentów.
- Zróżnicowane czasy wykonania rozkazów - od jednego do wielu cykli zegara.
- Układ sterowania procesora jest przeważnie mikroprogramowany.

# RISC

Badania statystyczne wykonane w latach 80-tych nad stopniem wykorzystania instrukcji CISC w typowych programach, pisanych przez programistów lub generowanych przez kompilator, wykazały, że pokaźny procent skomplikowanych rozkazów CISC (70%) nie jest wykorzystywany i największe wykorzystanie dotyczy prostych rozkazów. To spowodowało powstanie koncepcji architektury procesora o uproszczonej liście rozkazów.

# RISC

Procesory typu RISC mają następujące cechy architekuralne:

- Ograniczona lista rozkazów, zawierająca do 128 rozkazów wewnętrznych.
- Rozkazy wewnętrznych mają prostą treść operacyjną, realizują one osobno operacje dostępu do pamięci operacyjnej i operacje przetwarza danych w rejestrach.
- Mała liczba trybów adresowania dostępna w rozkazach wewnętrznych, do 4.
- Duża liczba rejestrów roboczych w procesorze, od 32 do 256.
- Mała liczba formatów rozkazów wewnętrznych, jednakowa długość słowa rozkazowego - często odpowiadająca pojedynczemu słowu.
- Ujednolicony czas wykonania rozkazów - od jednego do kilku cykli zegara.
- Układ sterowania procesora jest sprzętowy.

# Architektura procesorów CISC kontra RISC

	CISC			RISC	
	IBM 370/168	VAX 11/7	Intel 80486	Motorola 88000	MIPS R4000
Rok powstania	1973	1978	1989	1988	1991
Liczba rozkazów	208	303	235	51	94
Rozmiar rozkazu [B]	2-6	2-57	1 -11	4	32
Tryby adresowania	4	22	11	3	1
Liczba rejestrów roboczych	16	16	8	32	32
Rozmiar pamięci sterującej [kB]	420	480	246	-	-
Rozmiar pamięci podręcznej [KB]	64	64	8	16	128

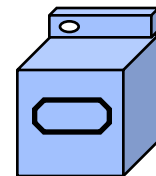
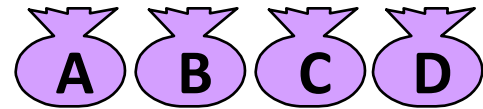
## Potokowość (*pipelining*)

- Technika budowy procesorów polegająca na podziale logiki procesora odpowiedzialnej za proces wykonywania programu (instrukcji procesora) na specjalizowane grupy w taki sposób, aby każda z grup wykonywała część pracy związanej z wykonaniem rozkazu.
- Grupy te są połączone sekwencyjnie – potok (*pipe*) – i wykonują pracę równocześnie, pobierając dane od poprzedniego elementu w sekwencji. W każdej z tych grup rozkaz jest na innym stadium wykonania.

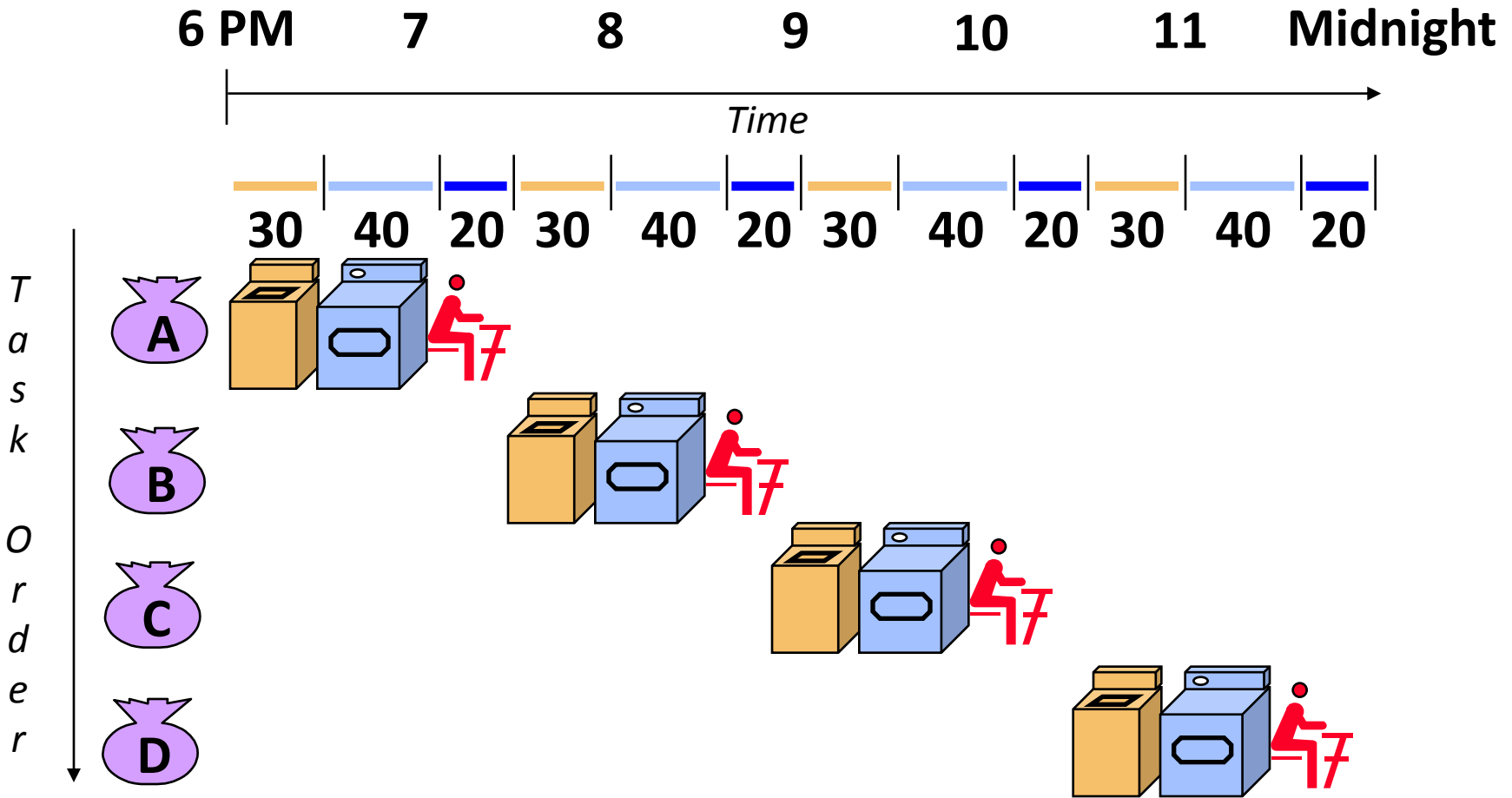


# Pipelining is Natural!

- Laundry Example
- Ann, Brian, Cathy, Dave each have one load of clothes to wash, dry, and fold
- Washer takes 30 minutes
- Dryer takes 40 minutes
- “Folder” takes 20 minutes

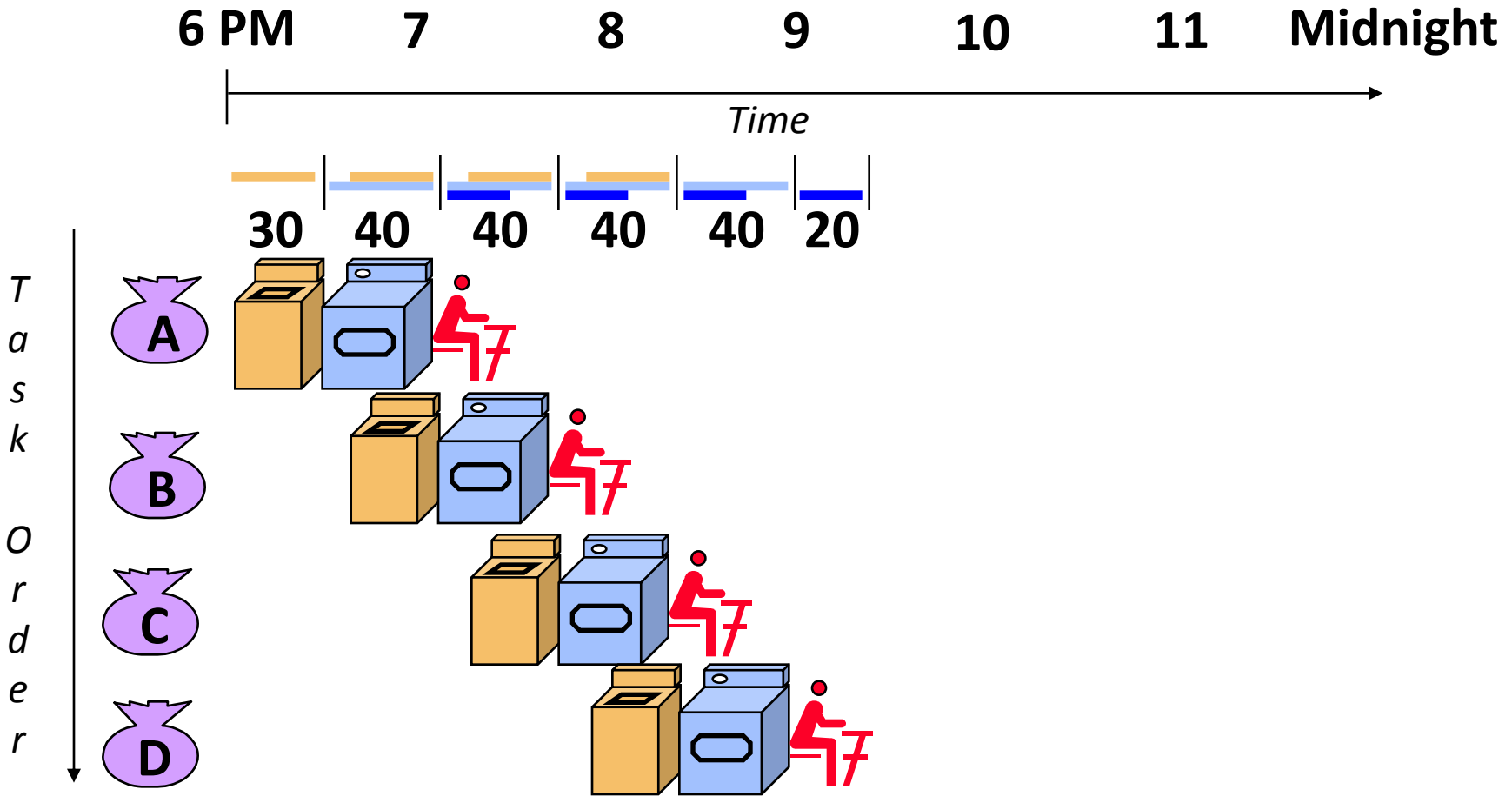


# Sequential Laundry



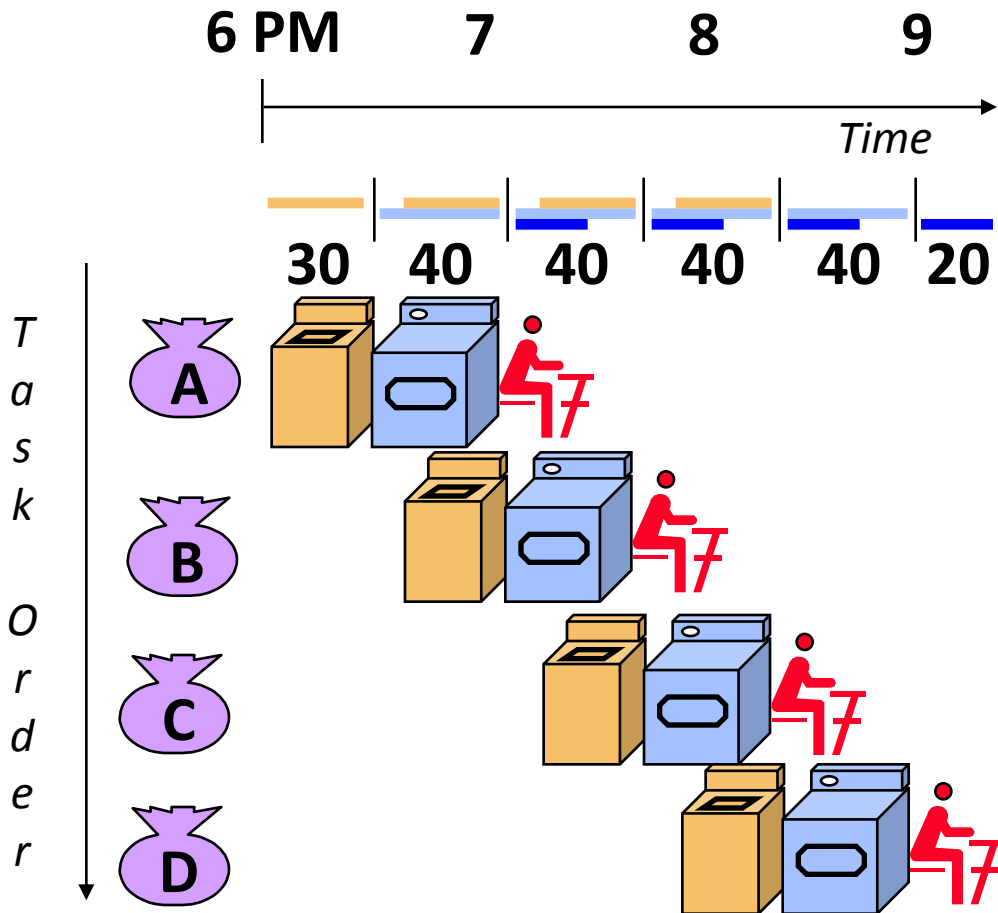
- Sequential laundry takes 6 hours for 4 loads
- If they learned pipelining, how long would laundry take?

# Pipelined Laundry: Start work ASAP



- Pipelined laundry takes 3.5 hours for 4 loads

# Pipelining Lessons



- Pipelining doesn't help **latency** of single task, it helps **throughput** of entire workload
- Pipeline rate limited by **slowest** pipeline stage
- **Multiple** tasks operating simultaneously using different resources
- Potential speedup = **Number pipe stages**
- Unbalanced lengths of pipe stages reduces speedup
- Time to “**fill**” pipeline and time to “**drain**” it reduces speedup
- Stall for Dependences

# Potokowość (*pipelining*)

- Pobranie instrukcji z pamięci – *instruction fetch* (IF)
- Zdekodowanie instrukcji – *instruction decode* (ID)
- Wykonanie instrukcji – *execute* (EX)
- Dostęp do pamięci – *memory access* (MEM)
- Zapisanie wyników działania instrukcji – *store; write back* (WB)



Części rozkazów oznaczone na zielono wykonywane są równocześnie.

# Potokowość (*pipelining*)

- W powyższym 5-stopniowym potoku, przejście przez wszystkie stopnie potoku (wykonanie jednej instrukcji) zabiera co najmniej 5 cykli zegarowych. Jednak ze względu na jednoczesną pracę wszystkich stopni potoku, jednocześnie wykonywanych jest 5 rozkazów procesora, każdy w innym stadium wykonania. Oznacza to, że taki procesor w każdym cyklu zegara rozpoczyna i kończy wykonanie jednej instrukcji i statystycznie wykonuje rozkaz w jednym cyklu zegara.
- Podstawowym mankamentem techniki potoku są rozkazy skoku, powodujące w najgorszym wypadku potrzebę przeczyszczenia całego potoku i wycofania rozkazów, które następowały zaraz po instrukcji skoku i rozpoczęcie zapełniania potoku od początku od adresu, do którego następował skok.
- Szacuje się, że taki skok występuje co kilkanaście rozkazów. Z tego powodu niektóre architektury programowe (np. [SPARC](#)) zakładały zawsze wykonanie jednego lub większej ilości rozkazów następujących po rozkazie skoku, tzw. skok opóźniony. Stosuje się także skomplikowane metody predykcji skoku lub metody programowania bez użycia skoków.

# Superskalarność

Jest to cecha mikroprocesorów oznaczająca możliwość jednoczesnego ukończenia kilku instrukcji w pojedynczym cyklu zegara. Jest to możliwe dzięki zwielokrotnieniu jednostek wykonawczych.

	IF	ID	EX	MEM	WB													
	IF	ID	EX	MEM	WB													
$i$		IF	ID	EX	MEM	WB												
		IF	ID	EX	MEM	WB												
$t$			IF	ID	EX	MEM	WB											
			IF	ID	EX	MEM	WB											
				IF	ID	EX	MEM	WB										
					IF	ID	EX	MEM	WB									
						IF	ID	EX	MEM	WB								
							IF	ID	EX	MEM	WB							

# Superskalarność

Pełne wykorzystanie wszystkich jednostek wykonawczych zależy od tego, czy w programie nie występują zależności między kolejnymi instrukcjami - tj. czy kolejna instrukcja jako argumentu nie potrzebuje wyników poprzedniego rozkazu.

Np. instrukcje

$$a = b + 5$$

$$c = a + 10$$

nie będą mogły zostać wykonane równoległe, ponieważ wartość  $c$  zależy od wyliczanego wcześniej  $a$ .

Jeśliby jednak usunąć zależność i napisać równoważnie

$$a = b + 5$$

$$c = b + 15$$

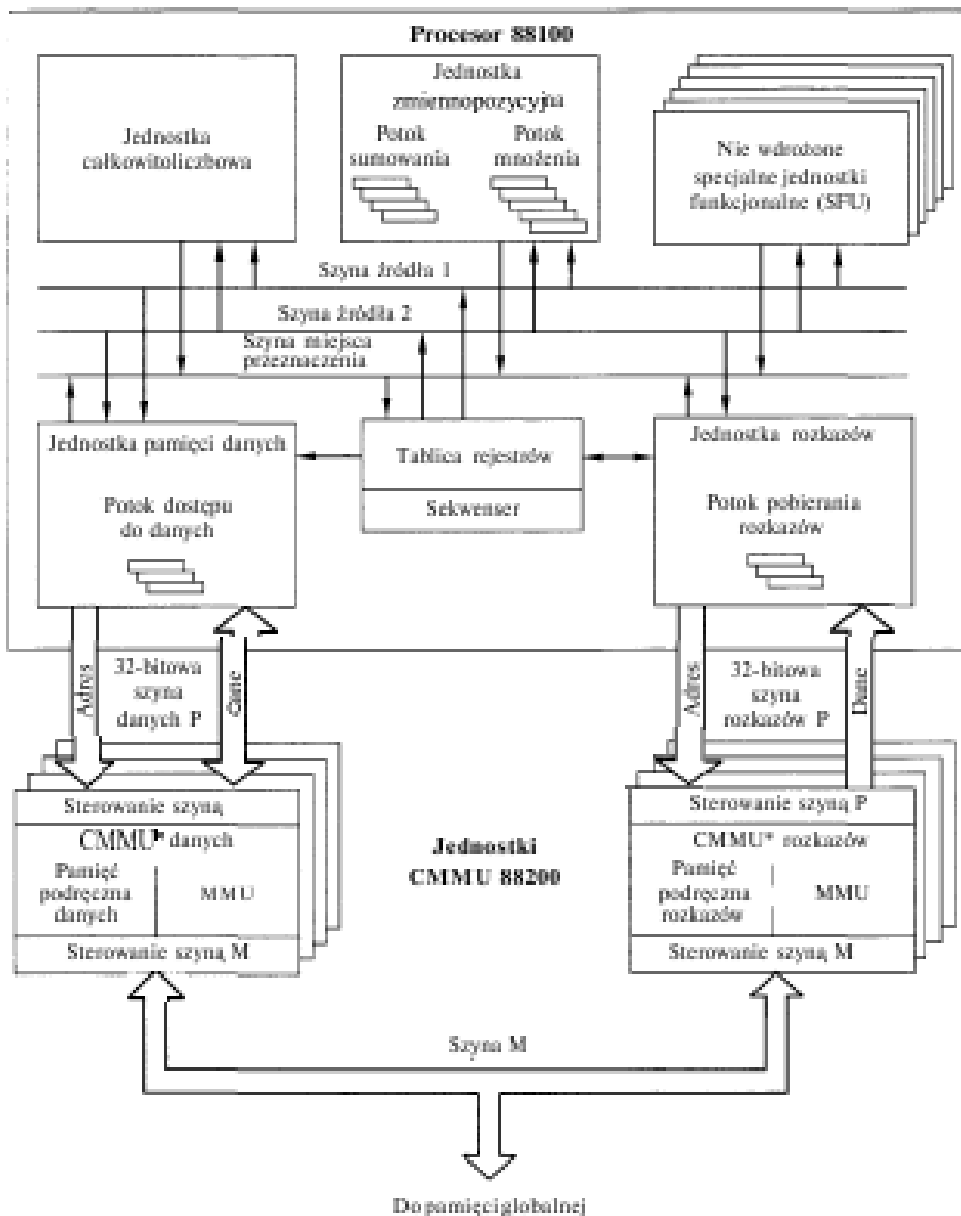
wykonywanie superskalarne będzie możliwe.

Minimalizacja zależności jest kluczowa, aby możliwe było pełne użycie dostępnych zasobów mikroprocesora. O właściwe rozmieszczenie instrukcji dba programista lub kompilator.

Ponadto współczesne procesory, np. Pentium Pro i nowsze, mogą zmieniać kolejność wykonywania instrukcji (zachowując oczywiście zależności między instrukcjami) - aby w pełni wykorzystać jednostki wykonawcze wyszukują instrukcje niezależne od siebie i wykonują je równoległe.



# Schemat blokowy systemu Motorola 88000 (RISC)



- układ procesora składa się z wielu niezależnych jednostek funkcyjnych połączonych z tablicą rejestrów
- Jednostki funkcyjne mogą działać niezależnie i współbieżnie

# Schemat blokowy systemu Motorola 88000 (RISC)

## **Jednostka całkowito liczbowa:**

Wykonuje operacje arytmetyczne na liczbach całkowitych, operacje na polach bitowych, operacje logiczne oraz dostępu do rejestru sterowania

## **Jednostka zmiennopozycyjna:**

5 etapowy potok sumatora i 6 etapowy potok mnożenia. Współbieżne wykonywanie wielu operacji zmiennopozycyjnych

## **Jednostka rozkazów:**

Odpowiedzialna za pobieranie rozkazów, ich dekodowanie i rozprowadzanie do jednostek wykonawczych

## **Jednostka pamięci danych:**

Odpowiedzialna za ładowanie i zapisywanie argumentów

# Pentium

- Pentium został pierwszym procesorem **CISC**, w którym użyto typowego dla konkurencyjnej architektury **RISC** rozwiązania zwanego "potokami" .
- Jeden potok "U" potrafiący wykonać każdą instrukcję, a drugi – "V" potrafiący wykonywać jedynie najprostsze, najczęściej używane komendy, co pozwalało Pentium na wykonywanie więcej niż jednej instrukcji w czasie pojedynczego cyklu.
- Pierwsze połączenie architektury x86 i RISC sygnalizowało, że jest możliwe połączenie tych dwóch rozwiązań tworząc procesory "hybrydowe".
- De facto Pentium był logicznie dwoma i486 korzystającymi ze wspólnego zestawu rejestrów i magistrali, wykonującymi pojedynczy program. Czasy wykonania większości operacji były podobne z i486 (większość instrukcji w 1 takt), jednak procesor był w stanie wykonywać efektywnie 2 instrukcje równocześnie, o ile nie były one złożone i od siebie zależne. W praktyce działało się tak przez 20-30% czasu przy nieoptymalizowanym kodzie.

# Pentium

- 64-bitowa szyna danych. Wszystkie główne rejestry pozostały 32-bitowe, ale podwojono ilość informacji pobieranej z RAM-u.
- Zestaw instrukcji **MMX** – prosty zestaw instrukcji **SIMD** pomocny w obróbce aplikacji multimedialnych.
- Rozdzielenie cache na cache instrukcji i danych i podwojenie jego wielkości (2x 8kB i 2x 16kB w wersji MMX).
- Bufory zapisu zwiększające prędkość współpracy z cache i magistralą (dodatkowo podwojone w wersji MMX).
- Jednostka **branch prediction** do przewidywania skoków .
- Wyższa częstotliwość taktowania szyny (początkowo 60 i 66MHz).
- Przeprojektowany koprocesor (5-6x wydajniejszy niż w i486).
- Przy włączonym stronicowaniu dostępne były, obok 4kB, także strony o rozmiarze 4MB.
- Nowa architektura Pentium oferowała mniej więcej dwukrotnie większą moc obliczeniową w porównaniu z intelowskimi 486.

## **SIMD** (*Single Instruction, Multiple Data*)

- przetwarzanych jest wiele strumieni danych w oparciu o pojedynczy strumień rozkazów. Architektura SIMD jest charakterystyczna dla komputerów wektorowych.
- pierwsze komputery o architekturze SIMD stosowano głównie do obliczeń naukowo-technicznych (np. Geometric-Arithmetic Parallel Processor czy Thinking Machines CM-1 i CM-2).
- obecnie jednostki realizujące zadania zgodnie z metodologią SIMD obecne są także w stosowanych w domowych komputerach procesorach opartych o architekturę x86.

# MMX

*(MultiMedia eXtensions lub Matrix Math eXtensions)*

- to zestaw 57 instrukcji SIMD dla procesorów Pentium i zgodnych. Rozkazy MMX mogą realizować działania logiczne i arytmetyczne na liczbach całkowitych. Pierwotnie wprowadzone w 1997 przez Intela dla procesorów Pentium MMX.
- programy wykorzystujące rozkazy MMX były o wiele szybsze od analogicznych programów wykorzystujących zwykłe rozkazy procesora.
- MMX jest przeznaczony do szczególnych zastosowań, gdzie przetwarzane są duże ilości danych przez jeden określony algorytm – obróbka dźwięku i obrazu.
- obecnie w zwykłych programach komputerowych zastosowanie MMX jest praktycznie ograniczone. Stosowane są kolejne generacje rozkazów wektorowych SSE, SSE2.

# MMX

*(MultiMedia eXtensions lub Matrix Math eXtensions)*

Przykłady zastosowań:

- wyświetlanie grafiki trójwymiarowej;
- przekształcenia geometryczne, cieniowanie, teksturowanie;
- dekodowanie obrazów JPEG i PNG;
- dekodowanie i kodowanie filmów MPEG (m.in. wyznaczanie transformat DCT i IDCT);
- filtrowanie sygnałów: obrazów statycznych, filmów, dźwięku;
- wyświetlanie grafiki dwuwymiarowej (blue box, maskowanie, przezroczystość);
- wyznaczanie transformat: Haara, FFT.

# MMX

## Typy danych:

- MMX wprowadził nowe typy danych, w języku angielskim nazwane *packed*, czyli dosłownie spakowane, upakowane; w języku polskim lepszym terminem oddającym ich charakter jest wektor i macierz lub tablica. Owo "spakowanie" polega traktowaniu danych 64-bitowych jako składających z pewnej liczby odrębnych komórek o tej samej wielkości:
  - 8 × 8 bitów (*packed byte*),
  - 4 × 16 bitów (*packed word*),
  - 2 × 32 bity (*packed dword*),
  - 1 × 64 bity (*quad word*).
- Gdy wykonywane są działania na typach wektorowych ("spakowanych"), ta sama operacja wykonywana jest dla wszystkich komórek jednocześnie. Np. jeśli dodawane są dwa wektory 8 x 8 bitów, to **pojedynczy rozkaz** wykonuje osiem operacji dodawania danych 8-bitowych i zapisywane jest osiem wyników 8-bitowych.