Architektura komputerów

Wykład 4
Cykl rozkazowy procesora

Wojciech Kordecki

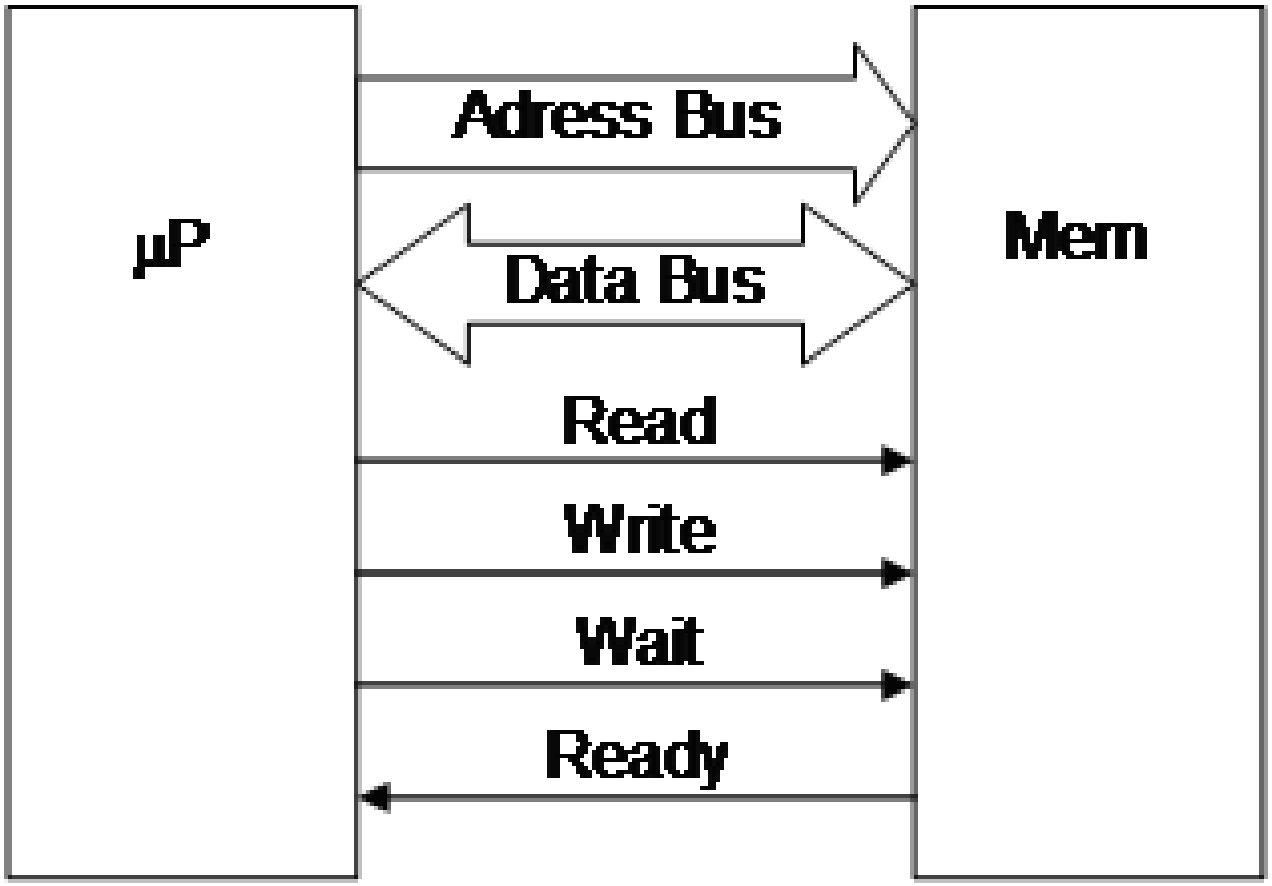
Collegium Witelona
Wydział Nauk Technicznych i Ekonomicznych
Zakład Informatyki

Semestr letni 2023/24

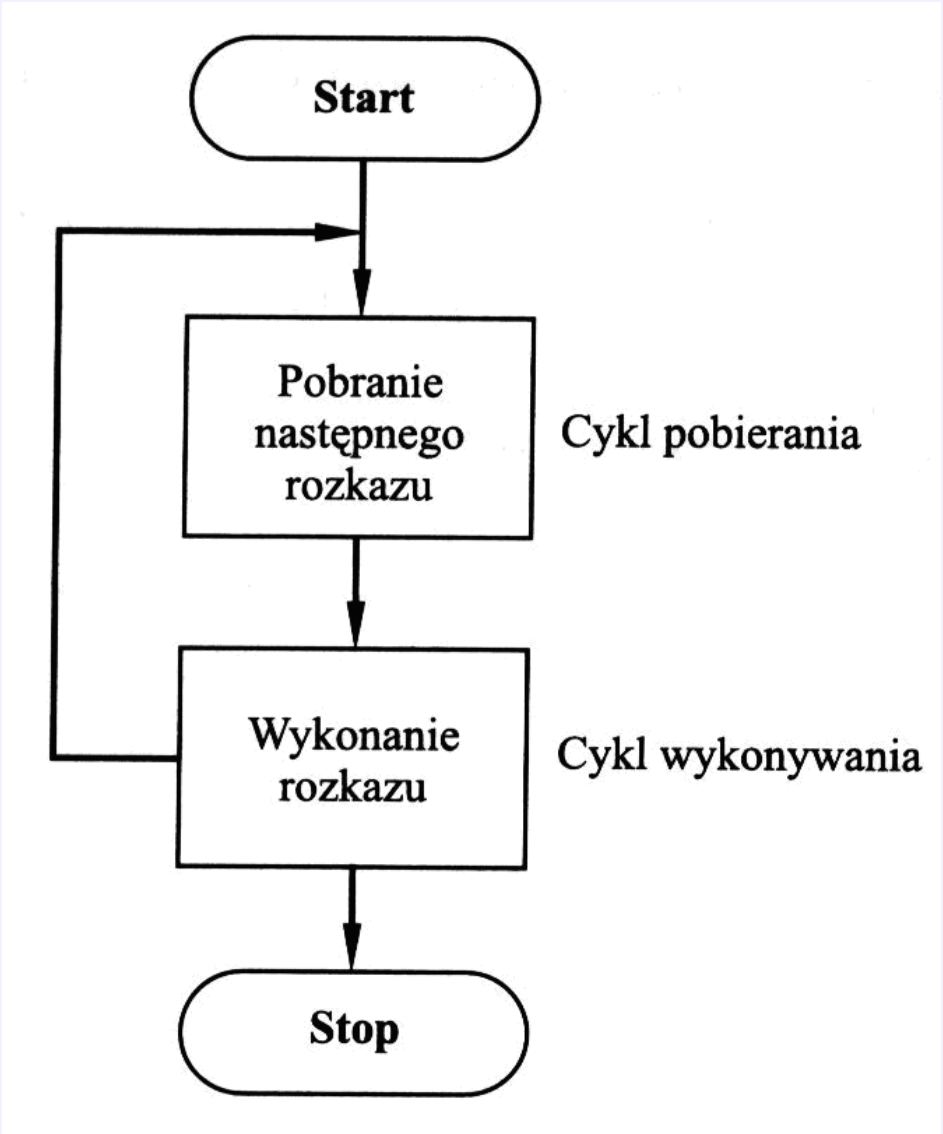
Wykonanie rozkazu

- Procesor pobiera z pamięci i wykonuje rozkazy w cyklu rozkazowym.
- Cykl rozkazowy składa się z cykli maszynowych, z których każdy jest związany z dostępem do pamięci (odczytem lub zapisem).
- Komunikacja pomiędzy procesorem a pamięcią odbywa się poprzez szynę adresową, szynę danych oraz sygnały sterujące odczytem i zapisem.

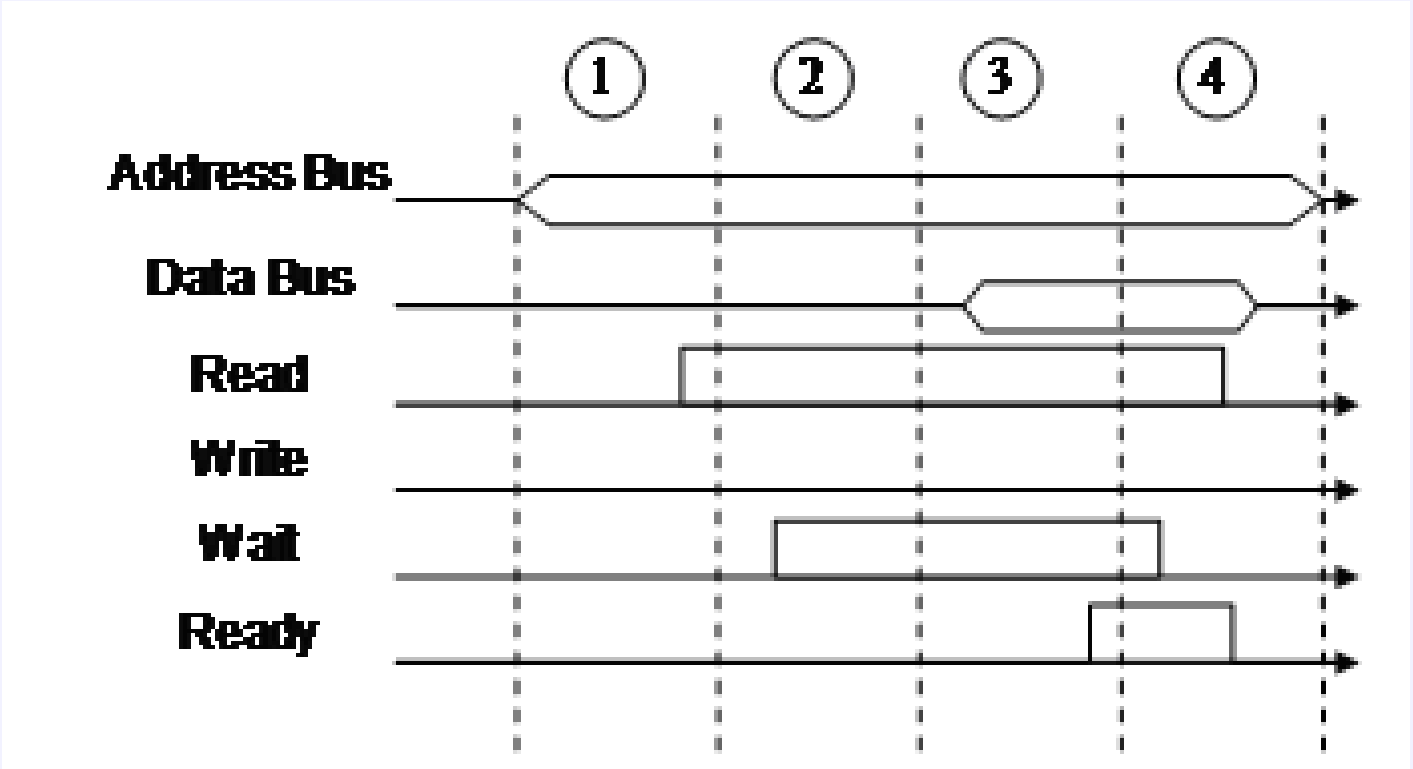
Komunikacja między procesorem i pamięcią



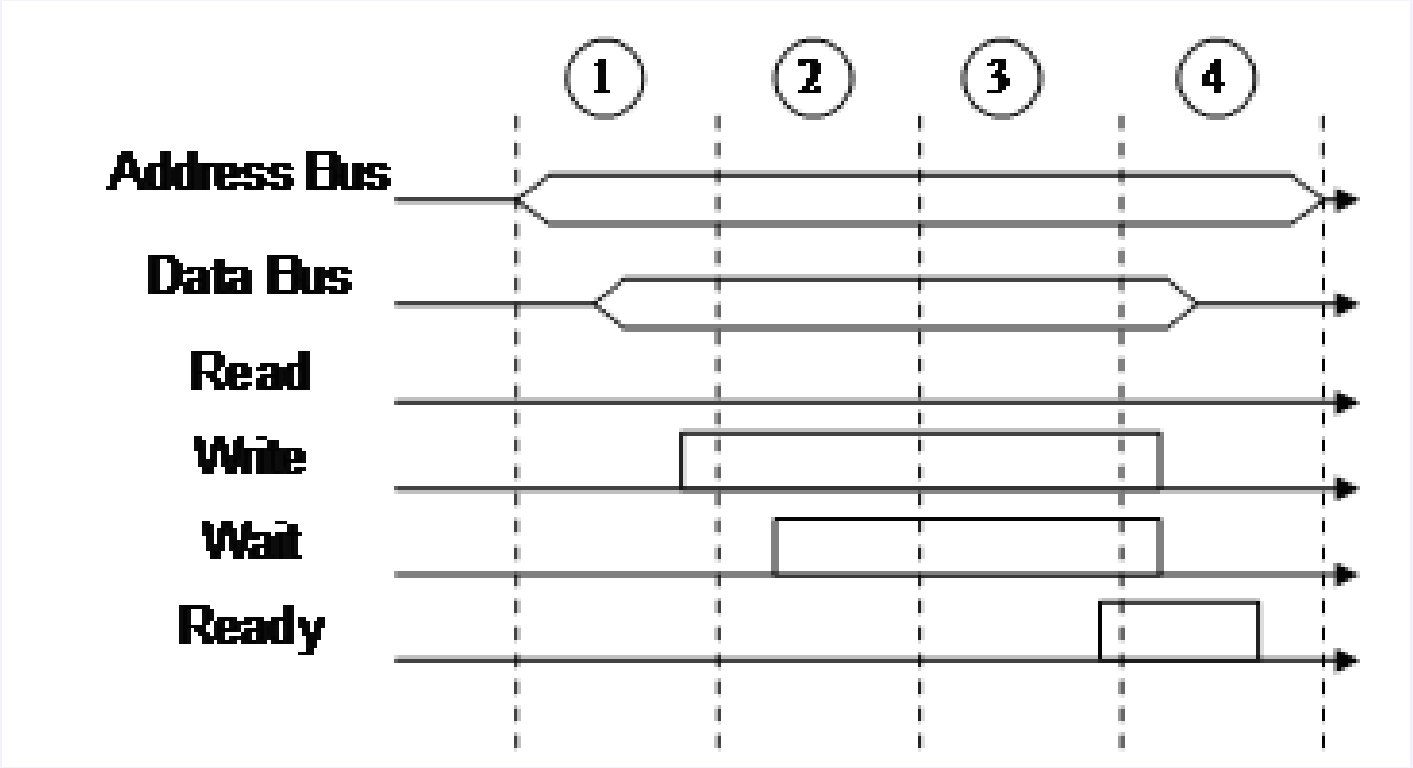
Schemat cyklu rozkazowego



Odczyt



Zapis



Cykl maszynowy w mikrokontrolerze 8051 (1)

Dwa cykle maszynowe:

s1	s2	s3	s4	s5	s6	s1	s2	s3	s4	s5	s6
Cykl maszynowy						Cykl maszynowy					

Cykl maszynowy w mikrokontrolerze 8051 (1)

Dwa cykle maszynowe:

s1	s2	s3	s4	s5	s6	s1	s2	s3	s4	s5	s6
Cykl maszynowy						Cykl maszynowy					

Typowy przebieg.

- S1 – odczyt (*fetch*): kod rozkazu,
- S2 – dekodowanie rozkazu i zwiększenie licznika rozkazów o 1,
- S3 – realizacja rozkazu,
- S4 – odczyt argumentu lub adresu bezpośredniego, ew. niewykorzystane.
- S5 –
- S6 –

Cykl maszynowy w mikrokontrolerze 8051 (1)

Dwa cykle maszynowe:

s1	s2	s3	s4	s5	s6	s1	s2	s3	s4	s5	s6
Cykl maszynowy						Cykl maszynowy					

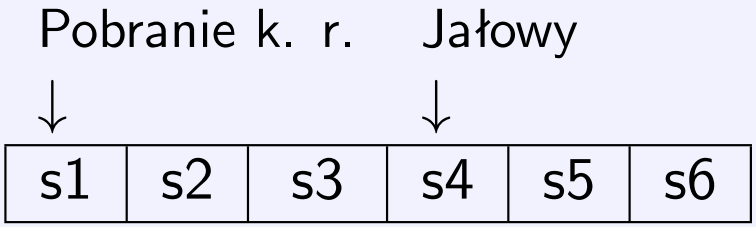
Typowy przebieg.

- S1 – odczyt (*fetch*): kod rozkazu,
- S2 – dekodowanie rozkazu i zwiększenie licznika rozkazów o 1,
- S3 – realizacja rozkazu,
- S4 – odczyt argumentu lub adresu bezpośredniego, ew. niewykorzystane.
- S5 –
- S6 –

Wykonanie w jednym lub dwóch cyklach maszynowych, tylko mnożenie i dzielenie w czterech.

Cykl maszynowy w mikrokontrolerze 8051 (2)

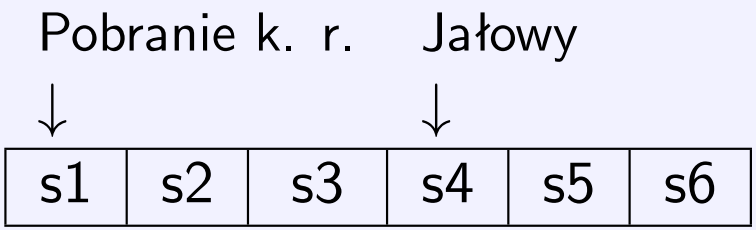
1 bajt, 1 cykl – np. `ADD A,R5`,



Kod $0010\,1r_1r_2r_r$: 0010 1101.

Cykl maszynowy w mikrokontrolerze 8051 (3)

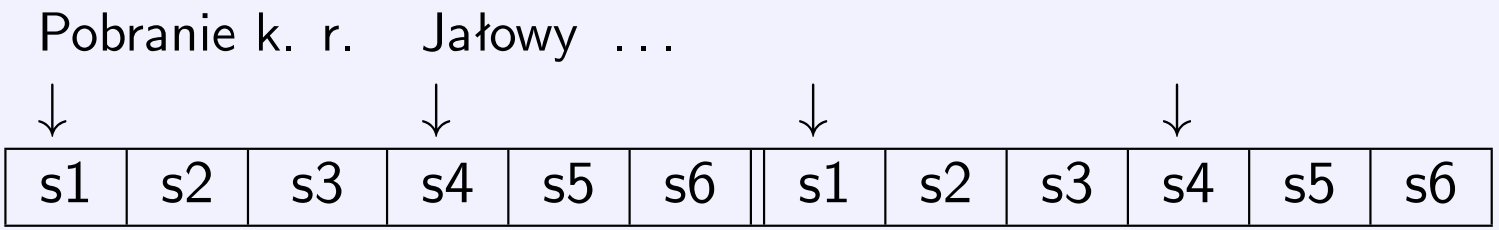
2 bajty, 1 cykl – np. `ADD A,#5`,



Kod 0010 0100 n: 0010 0100 0000 0101.

Cykl maszynowy w mikrokontrolerze 8051 (4)

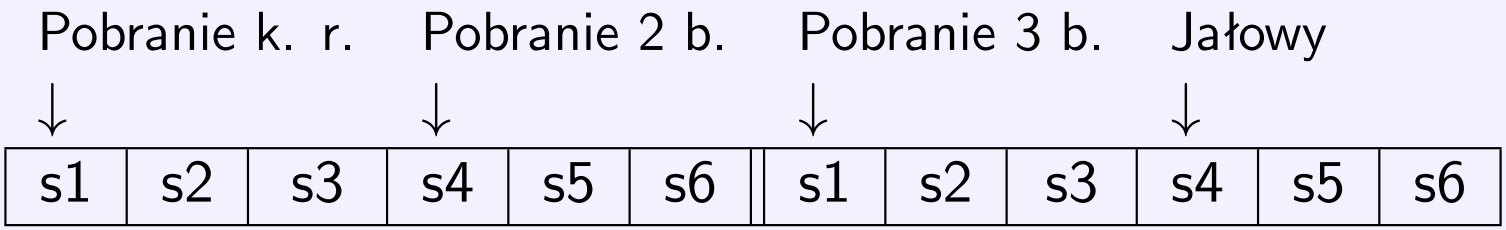
1 bajt, 2 cykle – np. `RET`,



Kod: 0011 0010.

Cykl maszynowy w mikrokontrolerze 8051 (5)

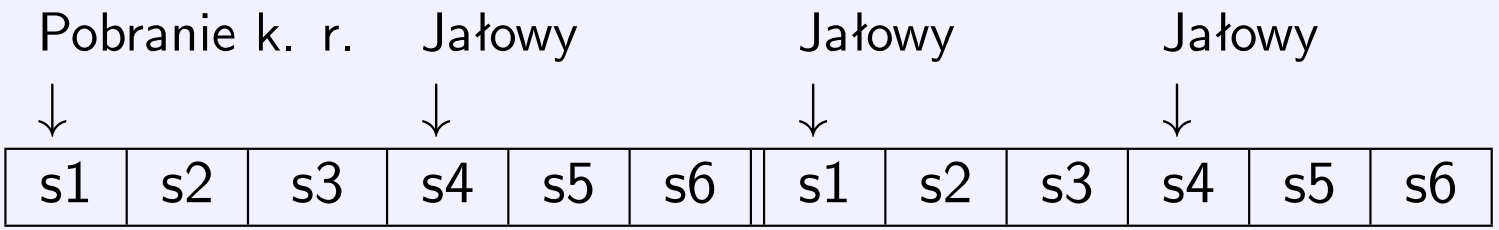
3 bajty, 2 cykle – np. `LJMP adr16`, np. $adr16 = 43\,825 = AB31\,H$



Kod $0000\,0010\,n_2\,n_1$: $000\,0010\,1010\,1011\,0011\,0001$.

Cykl maszynowy w mikrokontrolerze 8051 (6)

1 bajt, 2 cykle – np. `MOVX A,@DPTR`,



Kod 1110 0000.

Wady takiego systemu

Ograniczona szybkość

Szybkość działania procesora jest ograniczona wolnym dostępem do pamięci.

Taśma Forda

Zamiast gniazda – taśma.

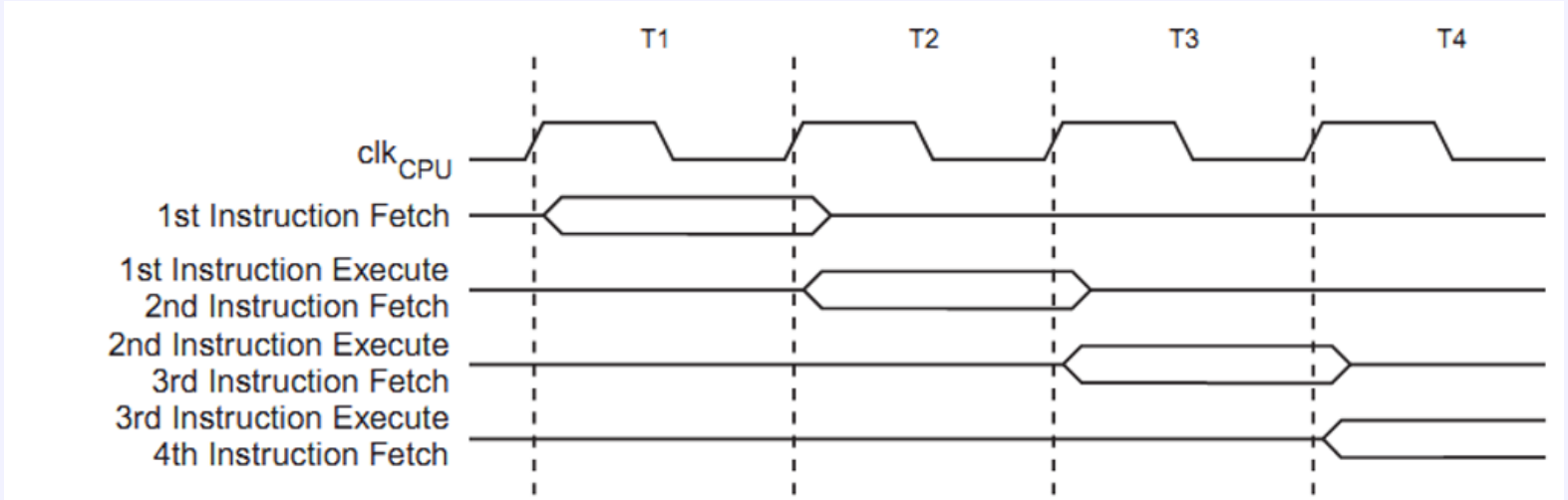
Taśma Forda

Zamiast gniazda – taśma.

Po wykonaniu pewnego fragmentu operacji, dalszy ciąg operacji na tym samym obiekcie jest wykonywany na następnym stanowisku.

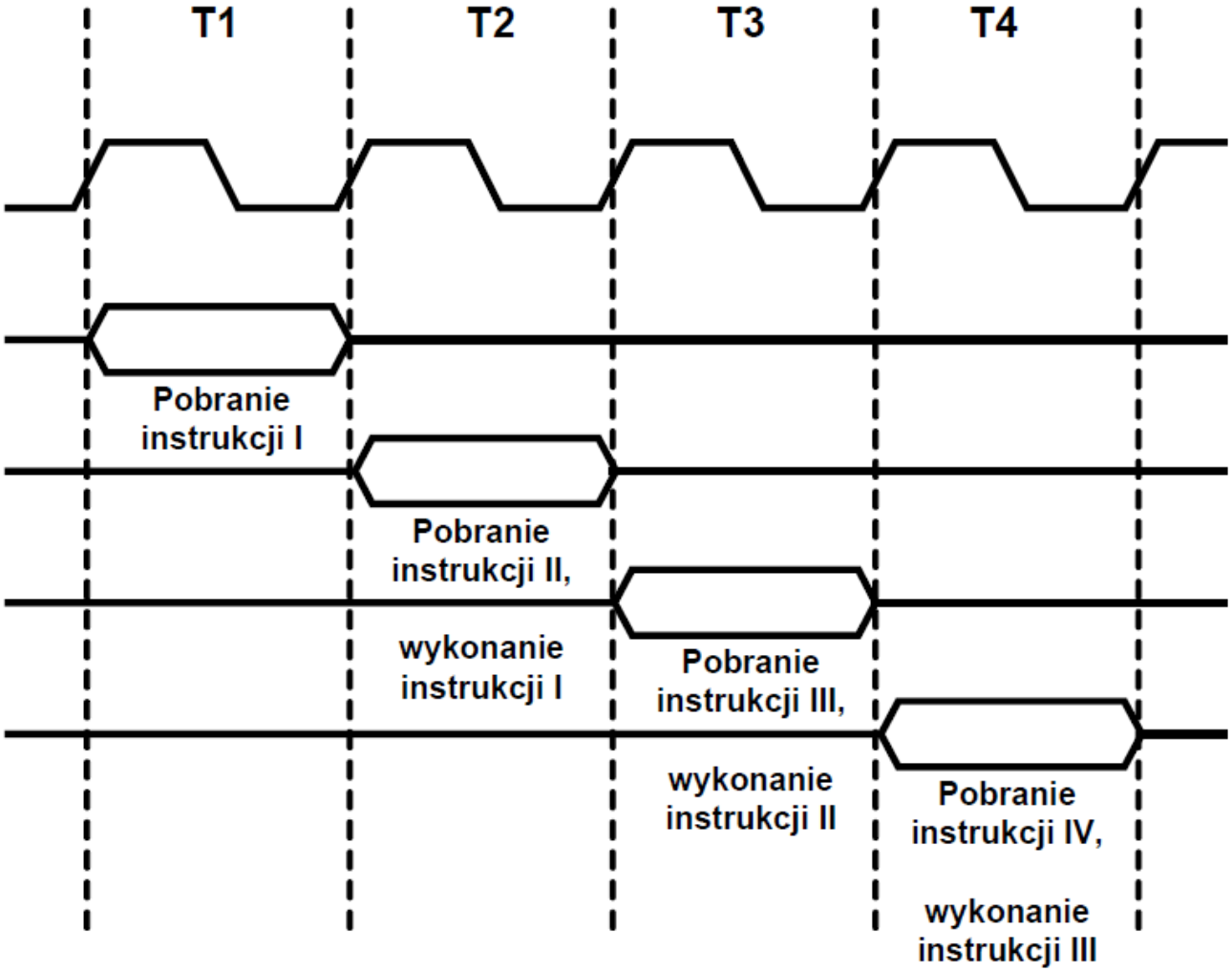
Cykl maszynowy w mikrokontrolerze AVR ATmega (1)

Architektura potokowa dwucyklowa: w pojedynczym cyklu wykonywana jest jedna instrukcja, a następna jest pobierana.



- Fetch – pobranie.
- Execute – wykonanie.

Cykl maszynowy w mikrokontrolerze AVR ATmega (2)



Intel Architecture 32 bit

32-bitowy model programowy mikroprocesora opracowany przez firmę Intel. Architektura IA-32 zaliczana jest z reguły do kategorii CISC, choć technologie wprowadzane stopniowo w nowszych wersjach procesorów IA-32 spełniają także wiele cech procesorów RISC.
Model IA-32 został wprowadzony w 1985 roku, wraz z procesorem Intel 80386 i jest najpopularniejszym modelem architektury stosowanym w komputerach ogólnego przeznaczenia. Stopniowo jest wypierany przez model 64-bitowy.

Instrukcje cyklu rozkazowego

Procesor wykonuje kolejne instrukcje w cyklu rozkazowym. Na cykl rozkazowy składa się:

- pobranie rozkazu,
- dekodowanie rozkazu,
- obliczenie adresu argumentu,odwołuje się do pamięci,
- obliczenie adresu fizycznego (binarnie na szynie adresowej) argumentu,
- wykonanie rozkazu,
- zapisanie wyniku,
- wyznaczenie położenia następnego rozkazu.

Współczesne procesory IA-32 (Intel Architecture-32) stosują przetwarzanie potokowe, czyli technologię polegającą na jednoczesnym wykonywaniu kolejnych etapów cyklu dla sąsiednich rozkazów przez różne bloki funkcjonalne.

Idea przetwarzania potokowego

Przetwarzanie potokowe = Pipeline.

r n =rozkaz numer n

Cykl	Faza	Faza	Faza	Faza	Faza	
1	r n	r $n-1$	r $n-2$	r $n-3$	r $n-4$	wynik $n-4$
2	r $n+1$	r n	r $n-1$	r $n-2$	r $n-3$	wynik $n-3$
3	r $n+2$	r n	r n	r n	r n	wynik $n-3$
4	r $n+3$	r $n+2$	r $n+1$	r n	r $n-1$	wynik $n-2$
5	r $n+4$	r $n+3$	r $n+2$	r $n+1$	r n	wynik $n-1$

Fazy

1. Pobranie (Prefetch, PF) poprzez pamięć podręczną L1 (ew. z pośredniej L2).

Fazy

1. Pobranie (Prefetch, PF) poprzez pamięć podręczną L1 (ew. z pośredniej L2).
2. Dekodowanie (Decode, DE) analizuje kod, ew. separuje przedrostki i argumenty. Następnie ew. oblicza się adres efektywny argumentów.

Fazy

1. Pobranie (Prefetch, PF) poprzez pamięć podręczną L1 (ew. z pośredniej L2).
2. Dekodowanie (Decode, DE) analizuje kod, ew. separuje przedrostki i argumenty. Następnie ew. oblicza się adres efektywny argumentów.
3. Wykonanie (Execute, EX) – fizyczny dostęp do pamięci dla pobrania argumentów i operacje na argumentach.

Fazy

1. Pobranie (Prefetch, PF) poprzez pamięć podręczną L1 (ew. z pośredniej L2).
2. Dekodowanie (Decode, DE) analizuje kod, ew. separuje przedrostki i argumenty. Następnie ew. oblicza się adres efektywny argumentów.
3. Wykonanie (Execute, EX) – fizyczny dostęp do pamięci dla pobrania argumentów i operacje na argumentach.
4. Zakończenie i zapisanie wyników (Write back, WB) – wynik z fazy EX jest zapisywany, przywracane są początkowe stany układów wewnętrznych (niewidocznych dla programisty), ustawiane są bity sygnalizacyjne.

Fazy

Inny podział i oznaczenia.

Fazy

Inny podział i oznaczenia.

1. Pobranie (Instruction fetch, IF)

Fazy

Inny podział i oznaczenia.

1. Pobranie (Instruction fetch, IF)
2. Dekodowanie (Instruction Decode, ID)

Fazy

Inny podział i oznaczenia.

1. Pobranie (Instruction fetch, IF)
2. Dekodowanie (Instruction Decode, ID)
3. Wykonanie (Execution, EX)

Fazy

Inny podział i oznaczenia.

1. Pobranie (Instruction fetch, IF)
2. Dekodowanie (Instruction Decode, ID)
3. Wykonanie (Execution, EX)
4. Zapis/odczyt do/z pamięci (Memory Read/Write, MEM)

Fazy

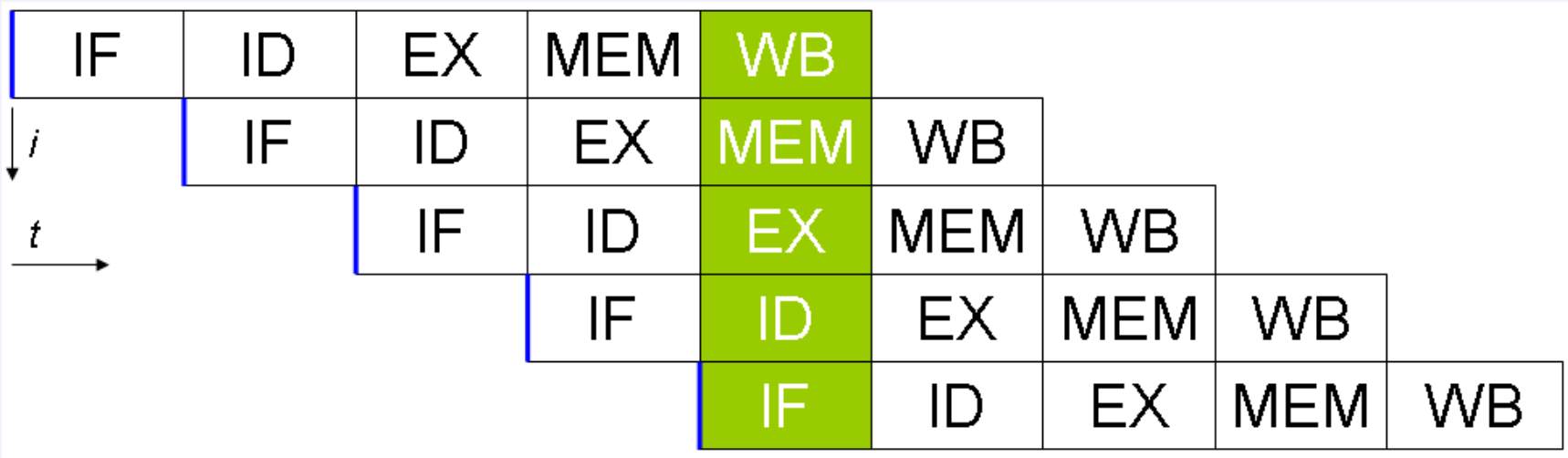
Inny podział i oznaczenia.

1. Pobranie (Instruction fetch, IF)
2. Dekodowanie (Instruction Decode, ID)
3. Wykonanie (Execution, EX)
4. Zapis/odczyt do/z pamięci (Memory Read/Write, MEM)
5. Zapis wyniku (Result Writeback, WB)

Jedna instrukcja w tym samym czasie

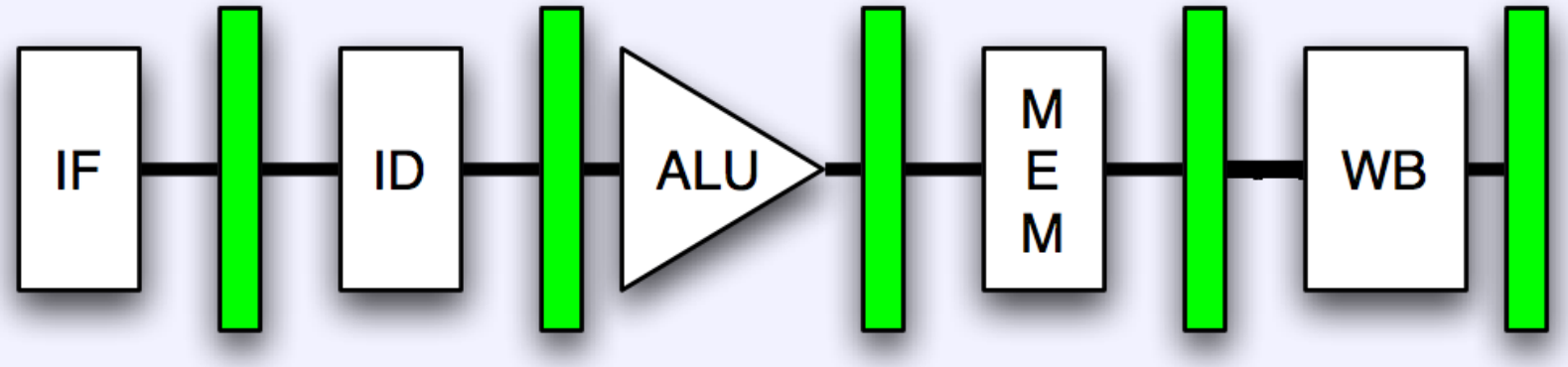
Źródło: Wikipedia

5 instrukcji w tym samym czasie



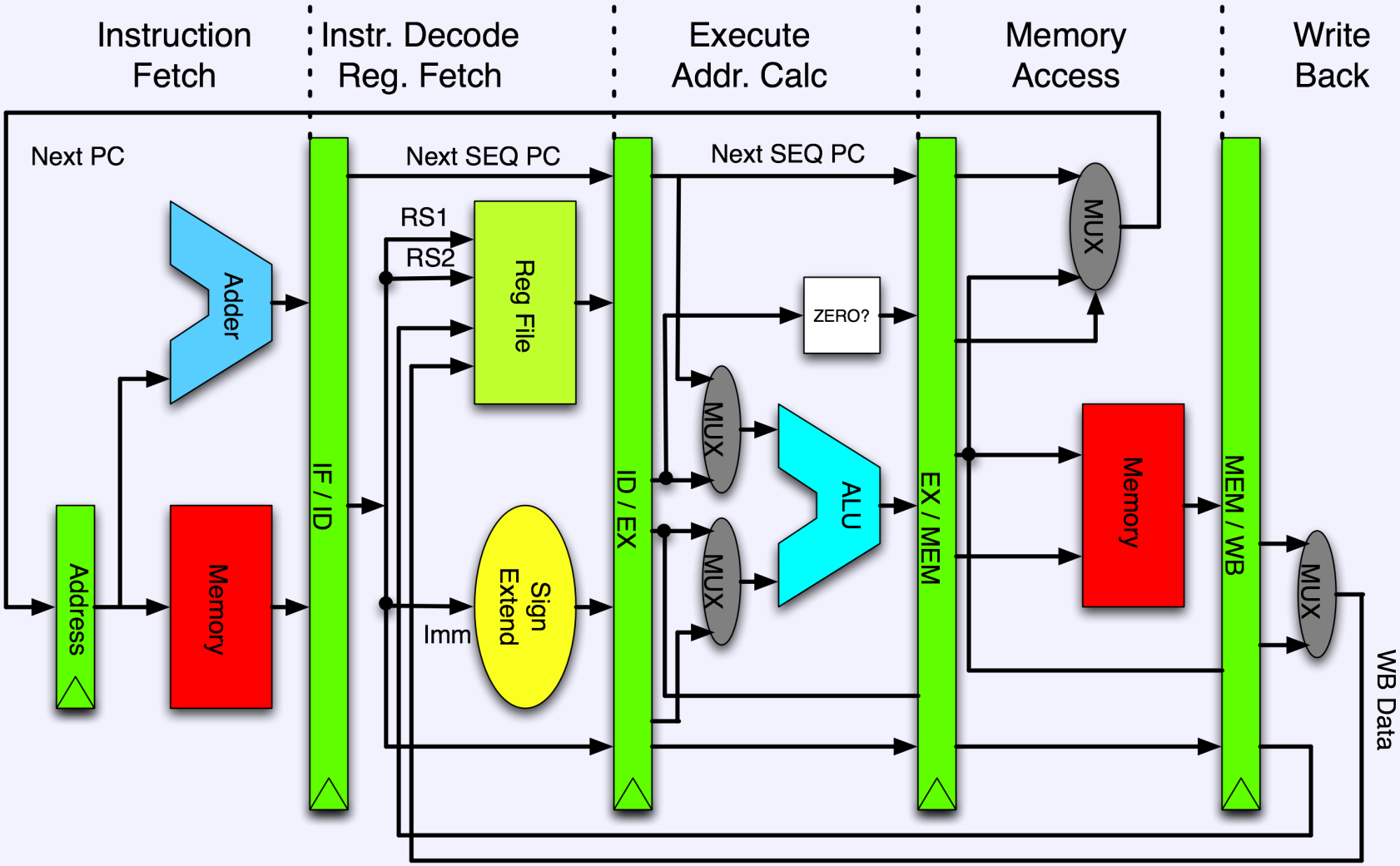
Źródło: Wikipedia

Schemat wykonania w procesorze



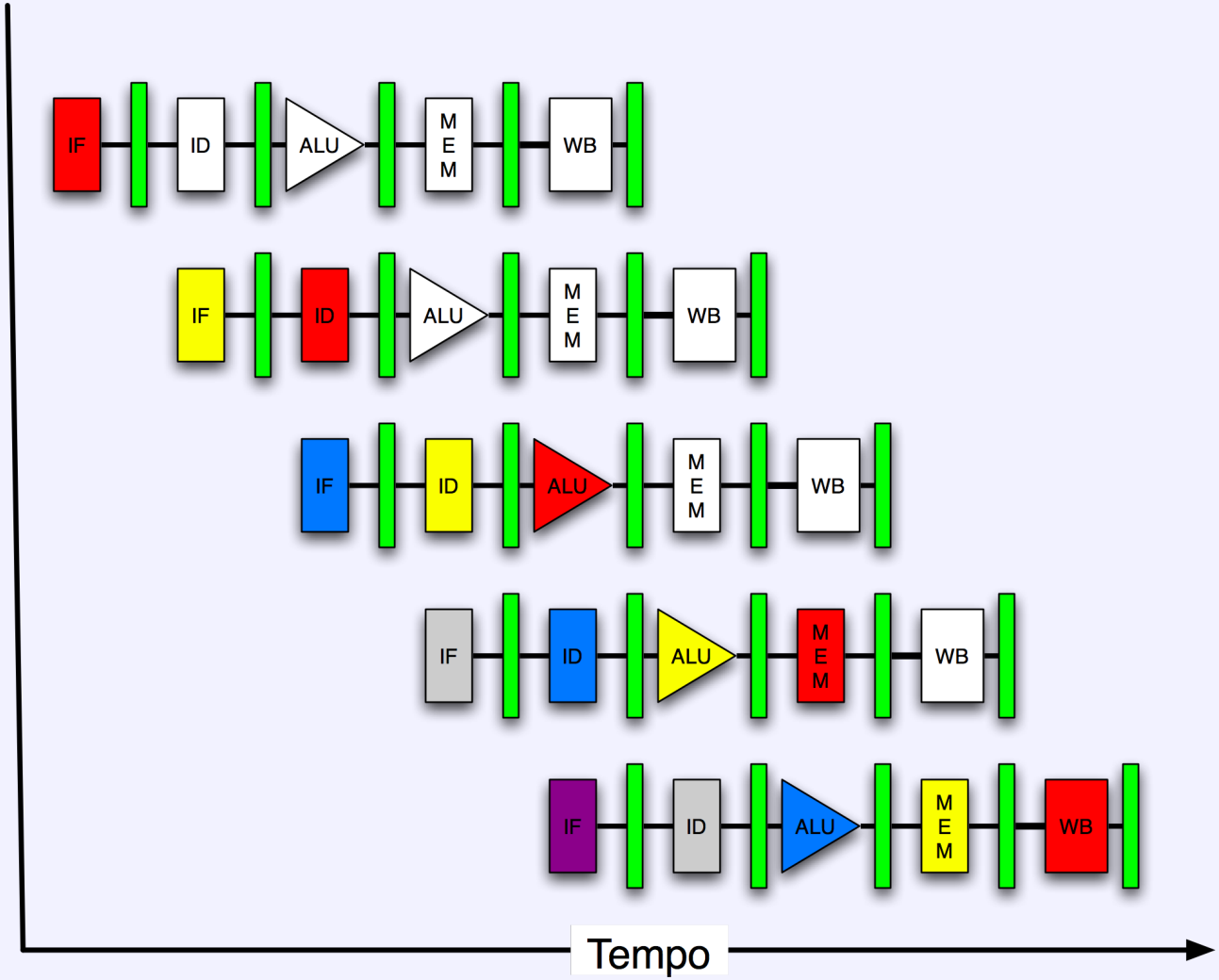
Źródło: Wikipedia

Procesor



Źródło: Wikipedia

Schemat wykonania w procesorze



Źródło: Wikipedia

Przykład

```
add ax,[bx] ;dodaj zawartość AX do zawartości komórki
            ; o adresie z BX, wynik do AX
```

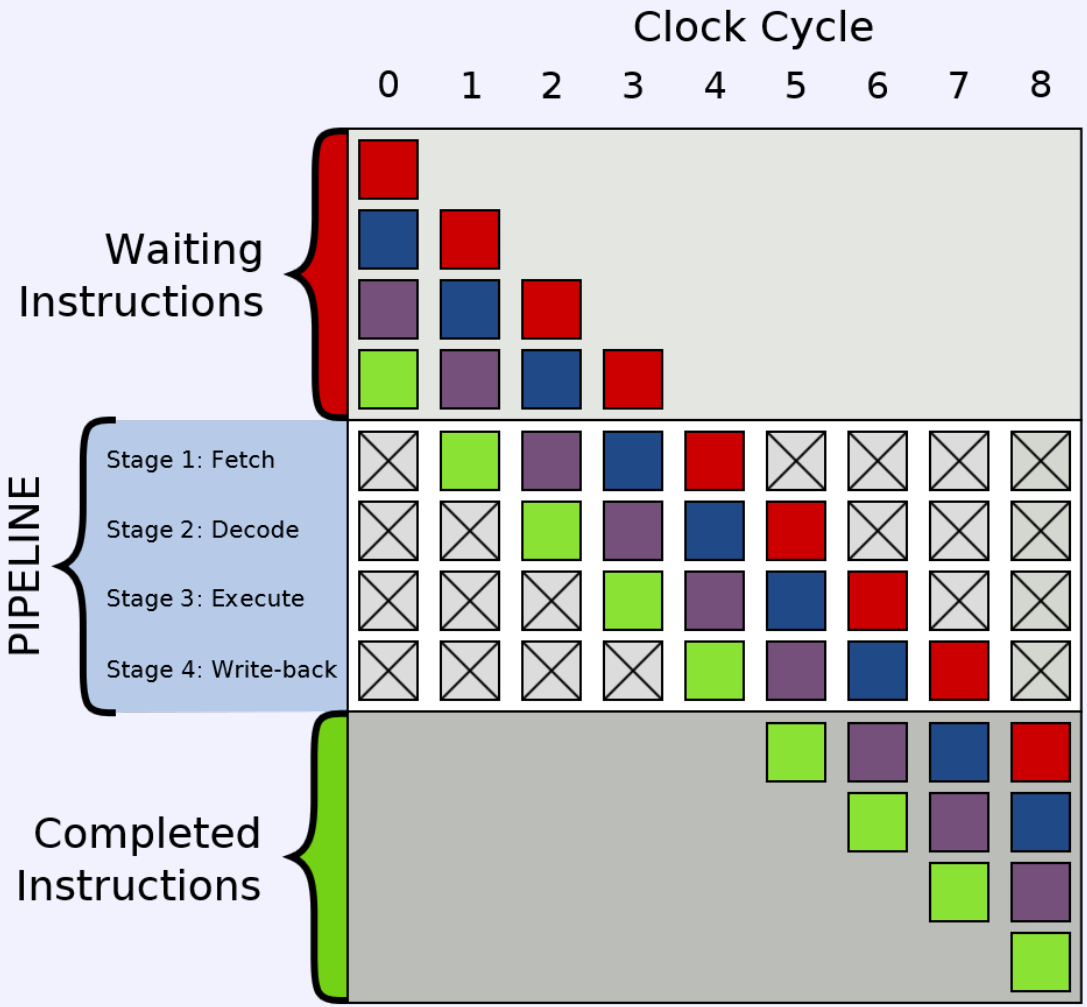
PF Pobranie kodu.

DE Określenie czynności do wykonania, obliczenie adresu $(16 \times DS) + BX$.

EX Dostęp do komórki o adresie $(16 \times DS) + BX$, operacja dodania do AX.

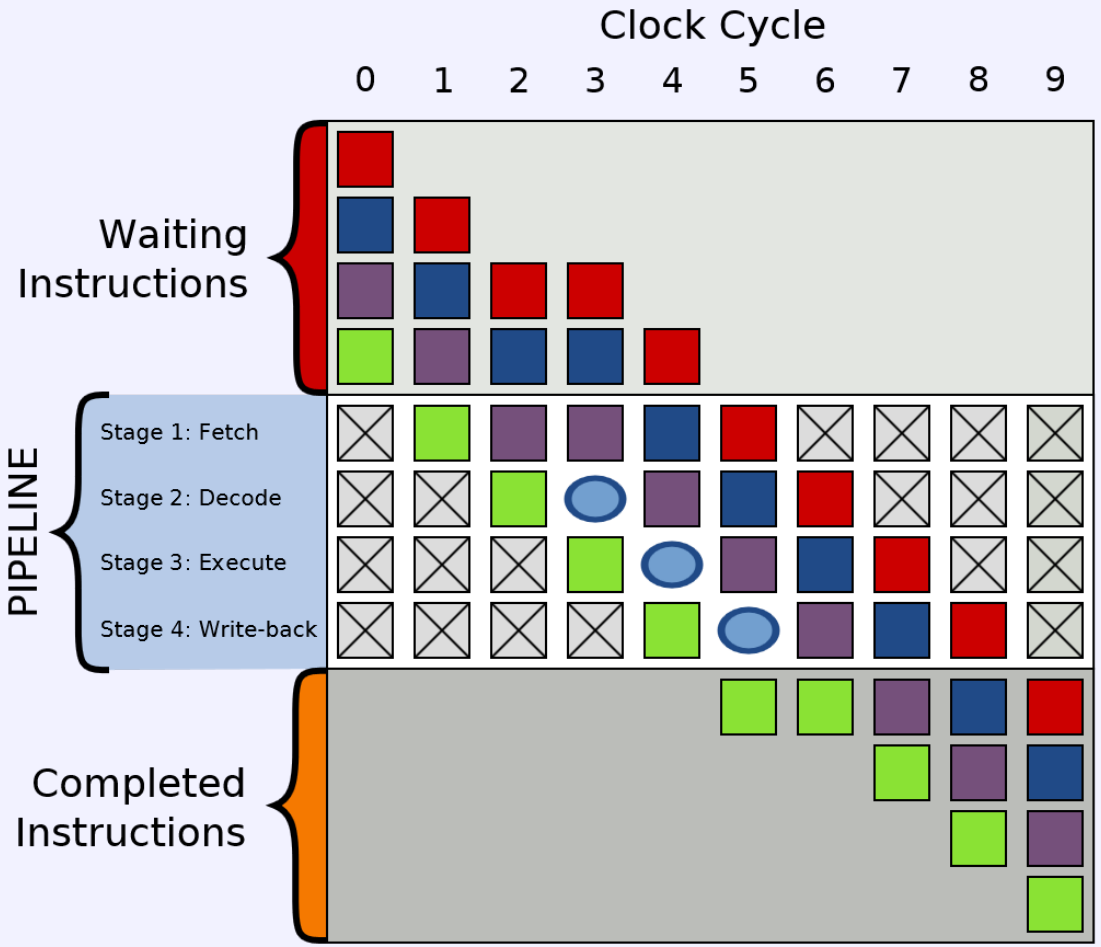
WB Umieszczenie wyniku w w AX, ustawienie flag (znaczników).

Cztery stany – bez opóźnień



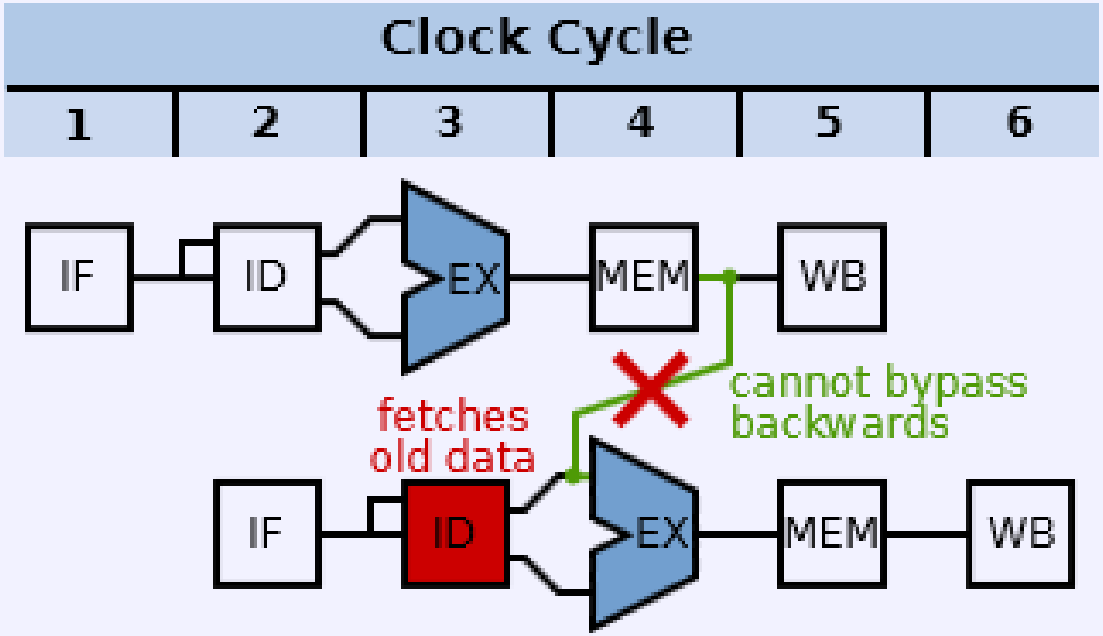
Źródło: Wikipedia

Cztery stany – opóźnienia (bąble)

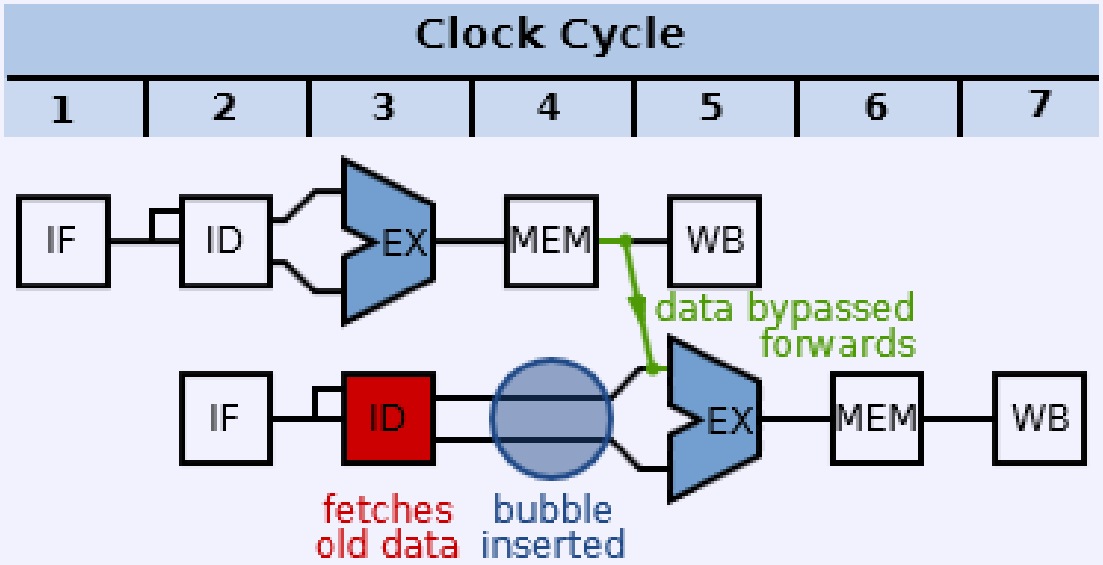


Źródło: Wikipedia

Opóźnienie w RISC



Opóźnienie w RISC – rozwiązanie



Źródło: Wikipedia

Procesory ARM

Architektura ARM (Advanced RISC Machine, pierwotnie Acorn RISC Machine) – 32-bitowa oraz 64-bitowa (Apple A7, 2013 r.) architektura (model programowy) procesorów typu RISC.

Różne wersje rdzeni ARM są szeroko stosowane w systemach wbudowanych i systemach o niskim poborze mocy, ze względu na ich energooszczędną architekturę.

ARM7 i wcześniejsze implementacje mają trzystanowy potok; są to *fetch*, *decode* oraz *execute*. Wyższe wersje, takie jak ARM9, mają głębsze potoki: Cortex-A8 ma 13 stanów.

Techniki superskalarne

Dwa potoki U i V oraz (jako osobny potok), FPU.

Techniki superskalarne

Dwa potoki U i V oraz (jako osobny potok), FPU.

Problemy:

Techniki superskalarne

Dwa potoki *U* i *V* oraz (jako osobny potok), FPU.

Problemy:

- zależności między potokami,

Techniki superskalarne

Dwa potoki *U* i *V* oraz (jako osobny potok), FPU.

Problemy:

- zależności między potokami,
- punkty rozgałęzień,

Techniki superskalarne

Dwa potoki *U* i *V* oraz (jako osobny potok), FPU.

Problemy:

- zależności między potokami,
- punkty rozgałęzień,
- korzystanie ze wspólnych rejestrów.

WAR

Problem.

```
mov bx,ax
add ax,cx
```

Najpierw trzeba odczytać AX, a dopiero potem przepisać CX do AX. Jest to uzależnienie typu WAR (*Write After Read*).

Przemianowywanie rejestrów

Podstawienie:

- AX $\Longrightarrow$ Reg0
- BX $\Longrightarrow$ Reg1
- CX $\Longrightarrow$ Reg2

Przemianowywanie rejestrów

Podstawienie:

- AX $\Longrightarrow$ Reg0
- BX $\Longrightarrow$ Reg1
- CX $\Longrightarrow$ Reg2

Przemianowanie

- AX $\Longrightarrow$ Reg3

Przemianowywanie rejestrów

Podstawienie:

- AX $\Longrightarrow$ Reg0
- BX $\Longrightarrow$ Reg1
- CX $\Longrightarrow$ Reg2

Przemianowanie

- AX $\Longrightarrow$ Reg3

Teraz równolegle:
Reg1:=Reg0 || Reg3:=Reg2+Reg0

Przewidywania skoków

Problem: co po instrukcji `if`?

Przewidywania skoków

Problem: co po instrukcji `if`?

Potoki muszą być stale napełniane, co może spowodować konieczność opróżnienia potoku.

Przewidywania skoków

Problem: co po instrukcji `if`?

Potoki muszą być stale napełniane, co może spowodować konieczność opróżnienia potoku.

Uwaga.

Ogromna strata czasu!

Przewidywania skoków

Problem: co po instrukcji `if`?

Potoki muszą być stale napełniane, co może spowodować konieczność opróżnienia potoku.

Uwaga.

Ogromna strata czasu!

Rozwiązania:

- Próbować przewidzieć dalszy bieg programu (*Branch Prediction*).
- W przypadku rozgałęzienia podążać w obydwu kierunkach (*Multiple Path Execution*).

Tablica dla jasnowidza

Tablica BTB (*Branch Target Buffer*).

Tablica dla jasnowidza

Tablica BTB (*Branch Target Buffer*).

W każdym wierszu tablicy informacje o jednym z punktów rozgałęzienia – zestaw adresów instrukcji skoków i adresów tych skoków. Dodatkowo status wiersza:
V:Valid oraz bity opisujące się zachowanie instrukcji: (H1, H2: *History Bits*).

Tablica dla jasnowidza

Tablica BTB (*Branch Target Buffer*).

W każdym wierszu tablicy informacje o jednym z punktów rozgałęzienia – zestaw adresów instrukcji skoków i adresów tych skoków. Dodatkowo status wiersza:
V:Valid oraz bity opisujące się zachowanie instrukcji: (H1, H2: *History Bits*).

Rozmiar	16	32	64	128	256	512	1024	2048
Trafionych[%]	40	50	65	72	78	80	85	87

Optymalizacja kodu

Dla znanego procesora tak pisać program, aby już w fazie doboru algorytmu i potem na etapie kompilacji, wykorzystać dobre strony konstrukcji, a ominąć słabe.

Optymalizacja kodu

Dla znanego procesora tak pisać program, aby już w fazie doboru algorytmu i potem na etapie kompilacji, wykorzystać dobre strony konstrukcji, a ominąć słabe.

Obecny trend, to zmniejszanie znaczenia optymalizacji kodu i przeniesienie problemu optymalizacji na procesor.

Źródła ważne

Przeszłość, teraźniejszość i przyszłość (stan do 2023 roku):

https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures

Od 8086 (1978) do Raptor (2022), Redwood Cove (2023), Crestmont (2023).

Szczegółowy podręcznik dla twórców kompilatorów i programistów w asemblerze (ostatnia zmiana 2022-06-08):

https://www.agner.org/optimize/microarchitecture.pdf

Tytuł

3.
The microarchitecture of Intel, AMD, and VIA CPUs
An optimization guide for assembly programmers and compiler makers

By Agner Fog. Technical University of Denmark.

O czym to jest?

1 Introduction

1.1 About this manual

This is the third in a series of five manuals:

1. Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms.
2. Optimizing subroutines in assembly language: An optimization guide for x86 platforms.
3. The microarchitecture of Intel, AMD and VIA CPUs: An optimization guide for assembly programmers and compiler makers.
4. Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs.
5. Calling conventions for different C++ compilers and operating systems.

Inne źródła

https://stackoverflow.com/questions/4830865/how-many-pipeline-stages-does-the-intel-core-i7-have

https://www.lkouniv.ac.in/site/writereaddata/siteContent/20200422161333844 5rohit_engg_pipelining_and_hazzard.pdf

https://www.nitsri.ac.in/Department/Electronics%20&%20Communication%20Engineering/COA-_Unit_3.pdf

Super książka

Książka trochę wiekowa, ale nadal aktualna w dużej części.

Jon Stokes, *Inside the Machine. An Illustrated Introduction to Microprocessor and Computer Architecture*. No Starch Press 2006.

Dostępna w internecie.