Organizacja i architektura komputerów [1]

Wykład 4

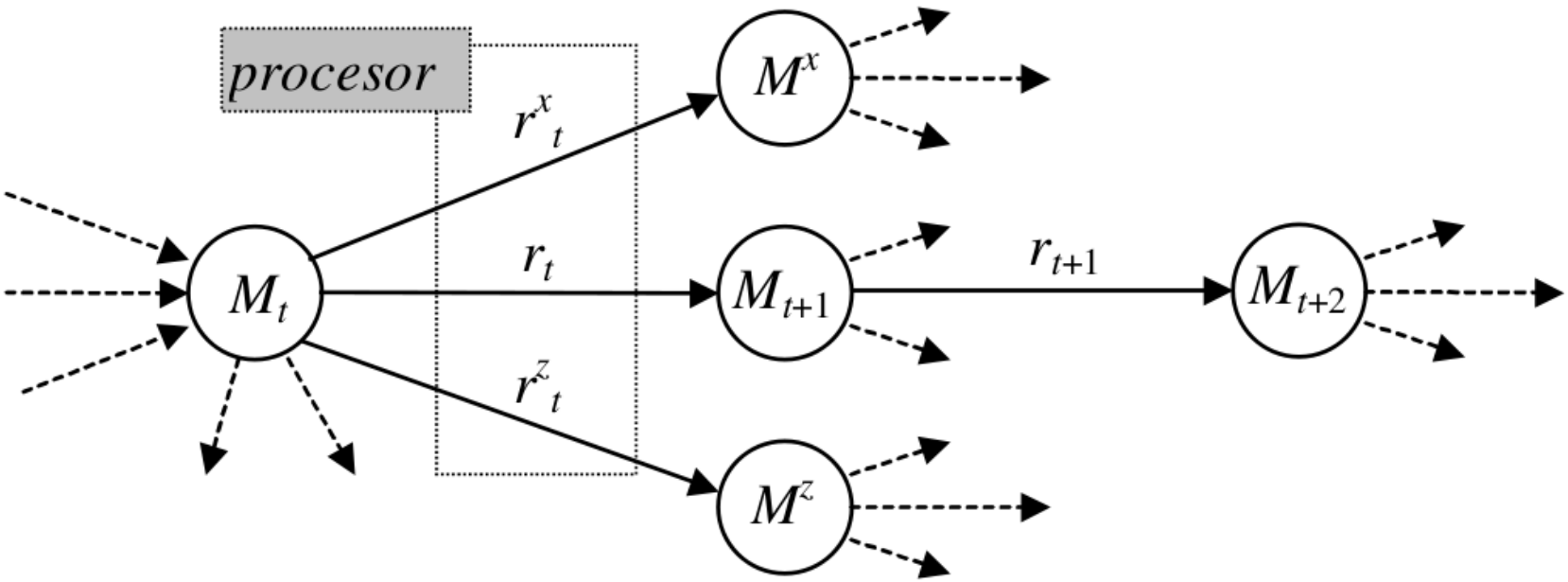
Piotr Patronik

15 marca 2016

[1] (Prawie) dokładna kopia slajdów dr hab inż. J. Biernata

Komputer jako automat (1)

- *Stan procesora* – zawartość rejestrów procesora **G**.
- *Stan komputera* – konfiguracja układów przechowujących informacje (storage), superpozycja *stanu procesora* **G** oraz *stanu pamięci* **S**
  $M_t = G_t \| S_t = \{g_j(t); j \in N\} \| \{s_i(t); i \in Z\} \in M$
- *Rozkaz* – funkcja $r_x \in R$, która odniesiona do *stanu początkowego* (*komputera*) wytwarza *stan wyjściowy komputera*
  $r_x : M \rightarrow M,\ r_x \in R,\ M_{t+1} = r_x(M_t) \in M$

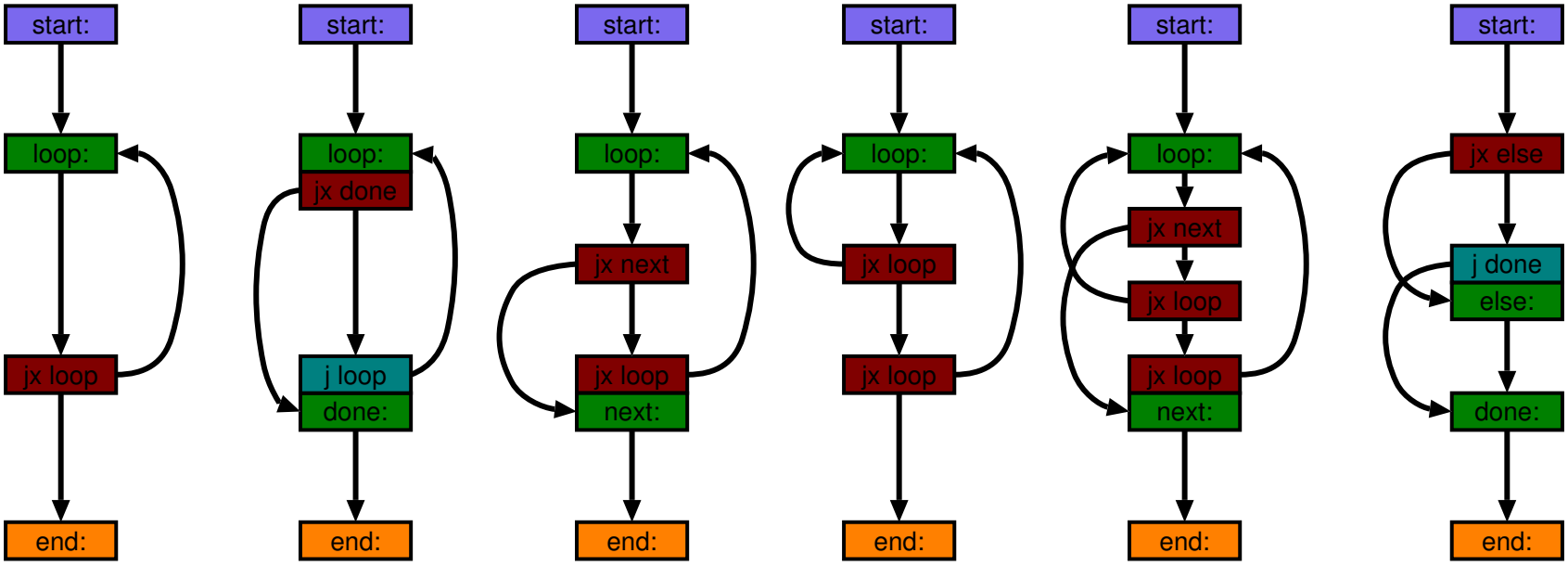


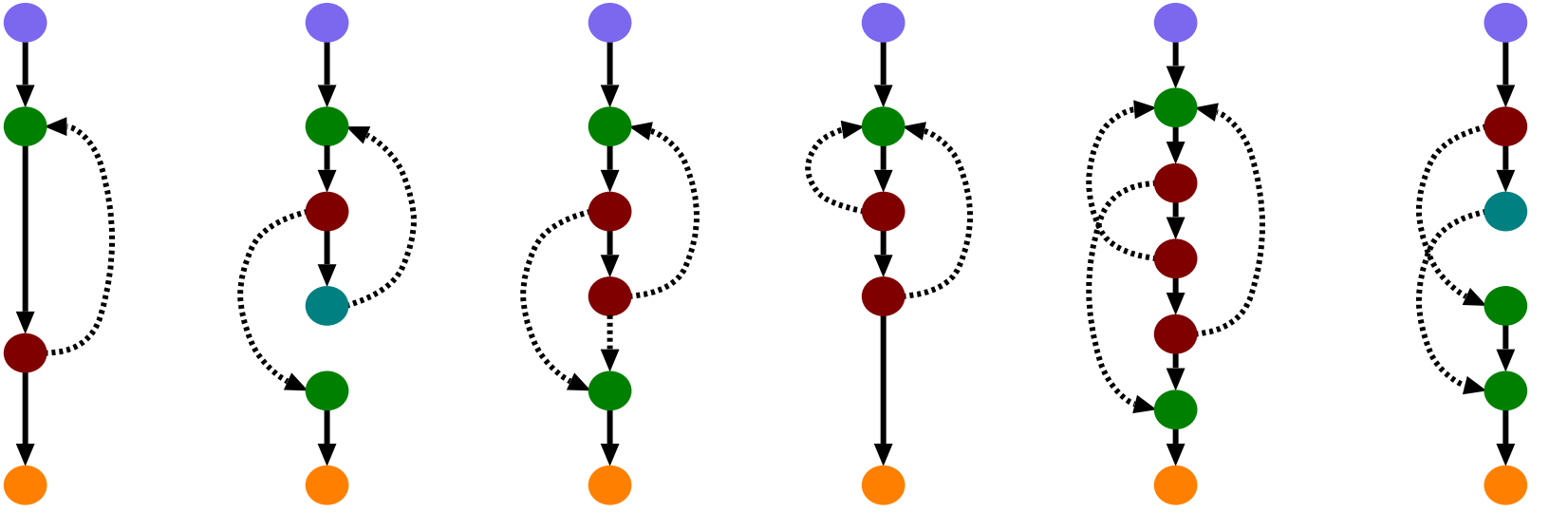
Komputer jako automat (2)

- *Pamięć* – dziedzina (domain) i zbiór wartości (range) rozkazów. W komputerze z programem zintegrowanym *treść* (działanie) *rozkazu* jest opisana przez *zawartość słowa* pamięci:
- Komputer z programem zintegrowanym może zmieniać kolejność wykonania instrukcji – modyfikować sekwencję sterowania (control *flow*). *Makrorozkaz* (proces) – superpozycja funkcji $r_x \in R$.
- *Architektura komputera* – zbiór rozkazów R i pamięć M.

Komputer jako automat (3)

Grafy sterowania i grafy przepływu danych





Komputer jako automat (4)

- Komputer to...
  - Automat
  - Układ przetwarzania danych (układ kombinacyjny)
- Funkcja: jądro obliczeniowe
  - Ścieżka sterowania (graf przepływu sterowania)
  - Ścieżka danych
- Blok podstawowy (basic block)
  - Ciąg rozkazów arytmetyczno/logicznych
  - `mov`, `add`, `mul`, `and` etc.
    (in-memory/in-cache processing)
  - Graf danych $(M, R) \rightarrow (M, R)$

Przetwarzanie wielowątkowe

Podstawowe koncepcje (cdn.)

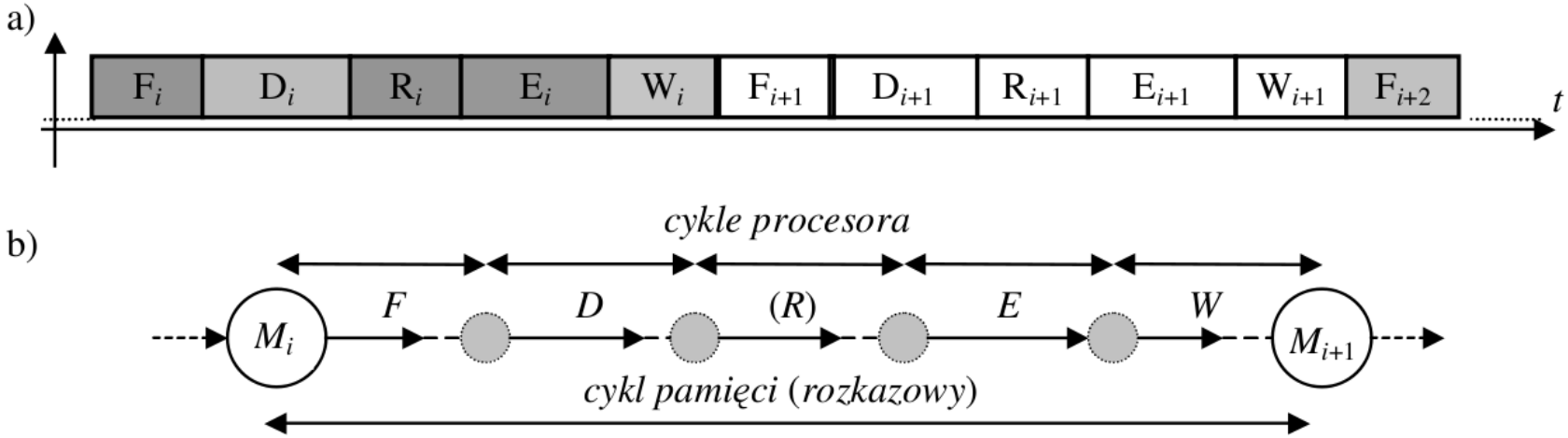
- .../PLP/TLP/ILP/ALP/...
- Jądro obliczeniowe
  - utrwalony graf sterowania
  - liczba egzemplarzy (liczba wątków)
  - wiele niezależnych ścieżek przebiegu
  - zbieżność ścieżek przebiegu (convergence)
  - spójność dostępów do pamięci (adresów)
  - czas wykonania
    - ścieżka krytyczna (l. powtórzeń pętli)
    - liczba jednostek wykonawczych
    - prawo Amdahla etc.

Komputer jako automat (2a)

- Z poziomu monitora (GDB)
  - Stan pamięci: `x` (`help x`)
  - Stan procesora: `i r` lub `i al`
  - Pojedynczy rozkaz: `si`
    - Pojedynczy rozkaz to w istocie wiele operacji

Cykl pamięci i cykle procesora

- Cykl – czas upływający pomiędzy kolejnymi zmianami stanu

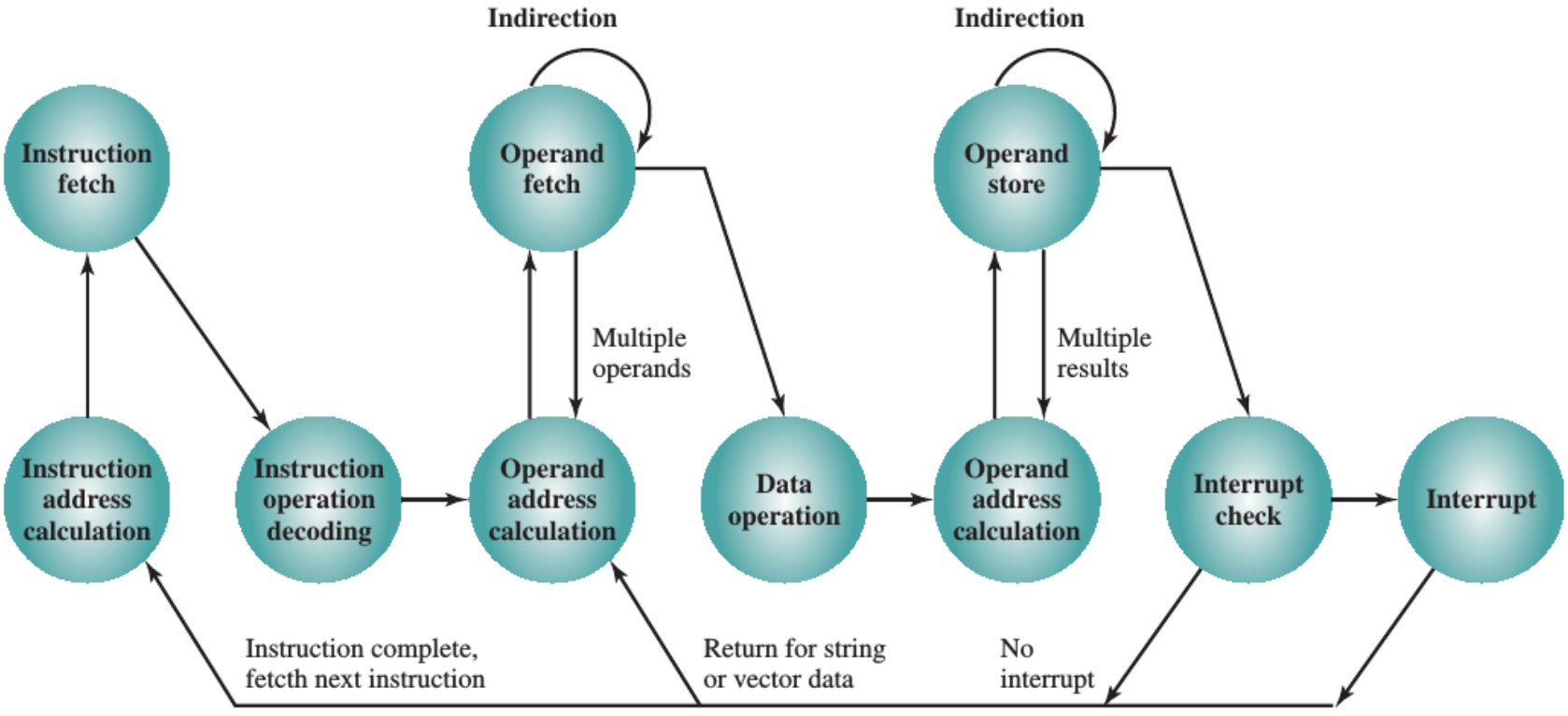


- Cykle (etapy) wykonania rozkazu a) diagram czasowy, b) graf automatu.

- Fazy (etapy) wykonania rozkazu:
- *F* (fetch) – pobranie kodu rozkazu z pamięci
- *D* (decode) – dekodowanie w celu wytworzenia sygnałów sterujących
- *R* (read) – pobranie operandu z pamięci (opcjonalne)
- *E* (execute) – wytworzenie wyniku (zmienny czas)
- *W* (write) – zwrotny zapis wyniku wykonania (do pamięci lub rejestru)

Cykl pamięci i cykle procesora (2)

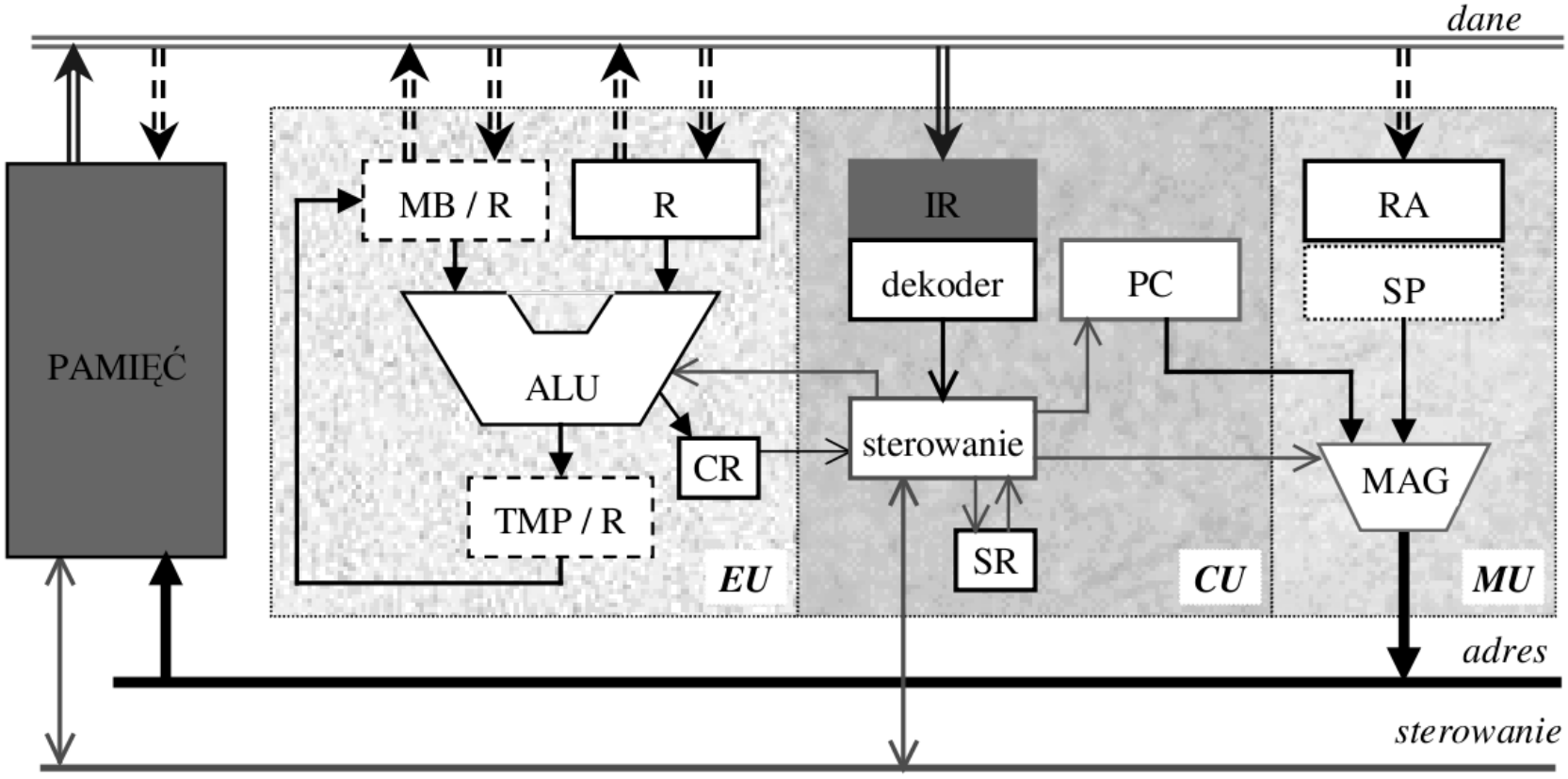
Cykl wykonania rozkazu



(Stallings)

Architektura procesora sekwencyjnego (F)

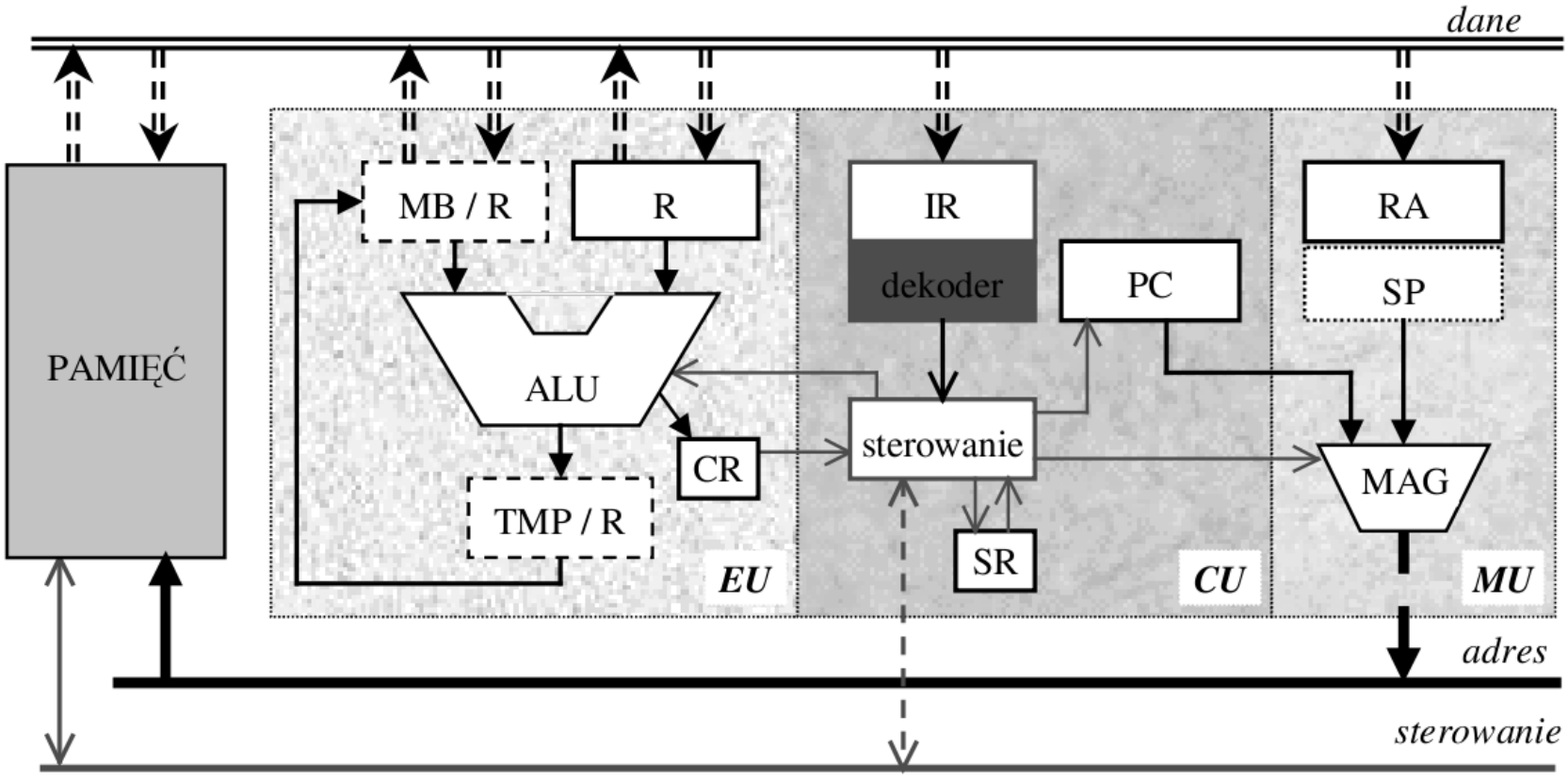
(F)etch



- *F* (pobranie kodu): adres (PC) $\longrightarrow$ pamięć $\longrightarrow$ rejestr rozkazów IR

Architektura procesora sekwencyjnego (D)

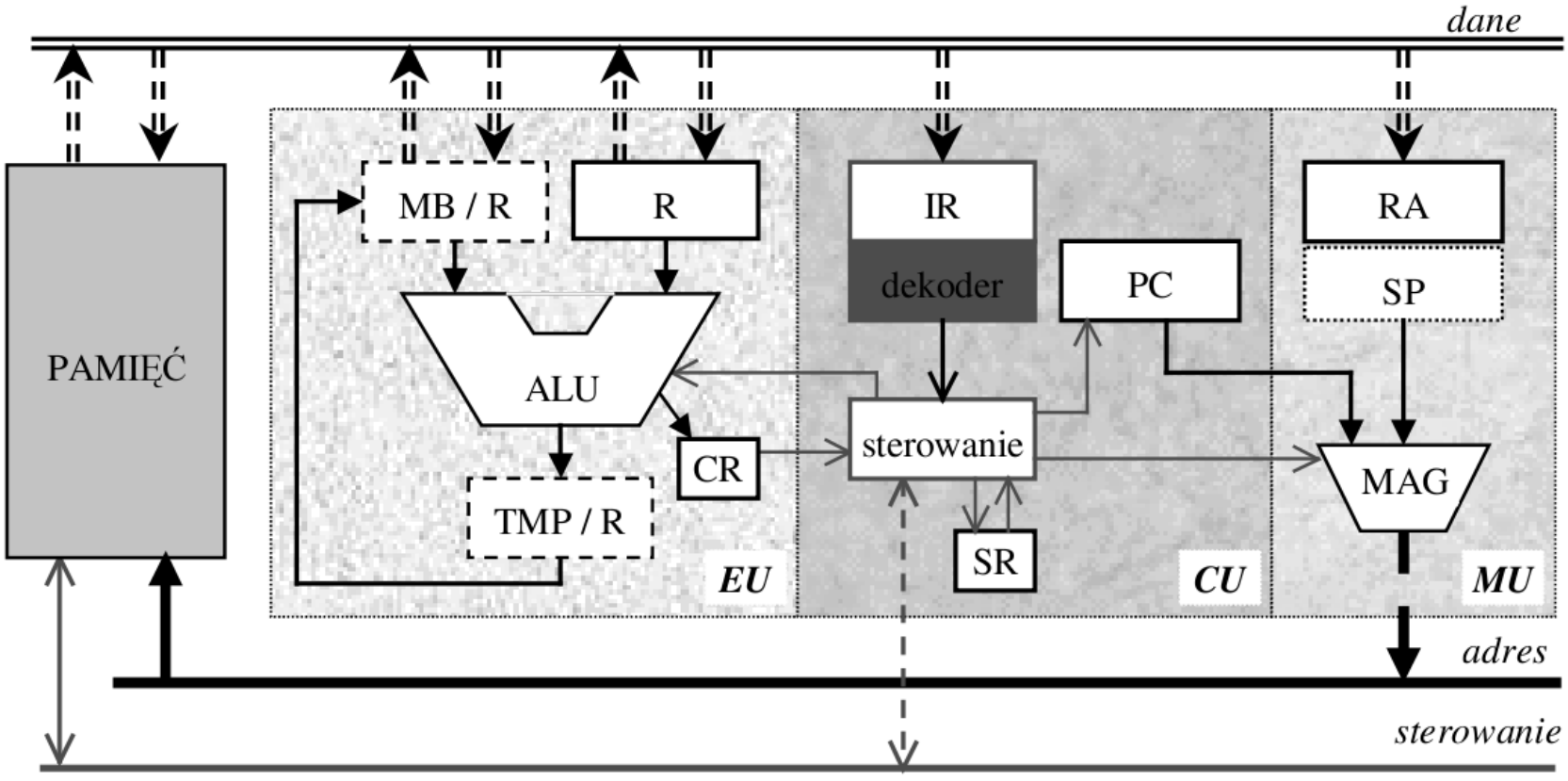
(D)ecode



- *D* (dekodowanie): rejestr rozkazów IR $\rightarrow$ sterowanie (CR,SR)

Architektura procesora sekwencyjnego (R)

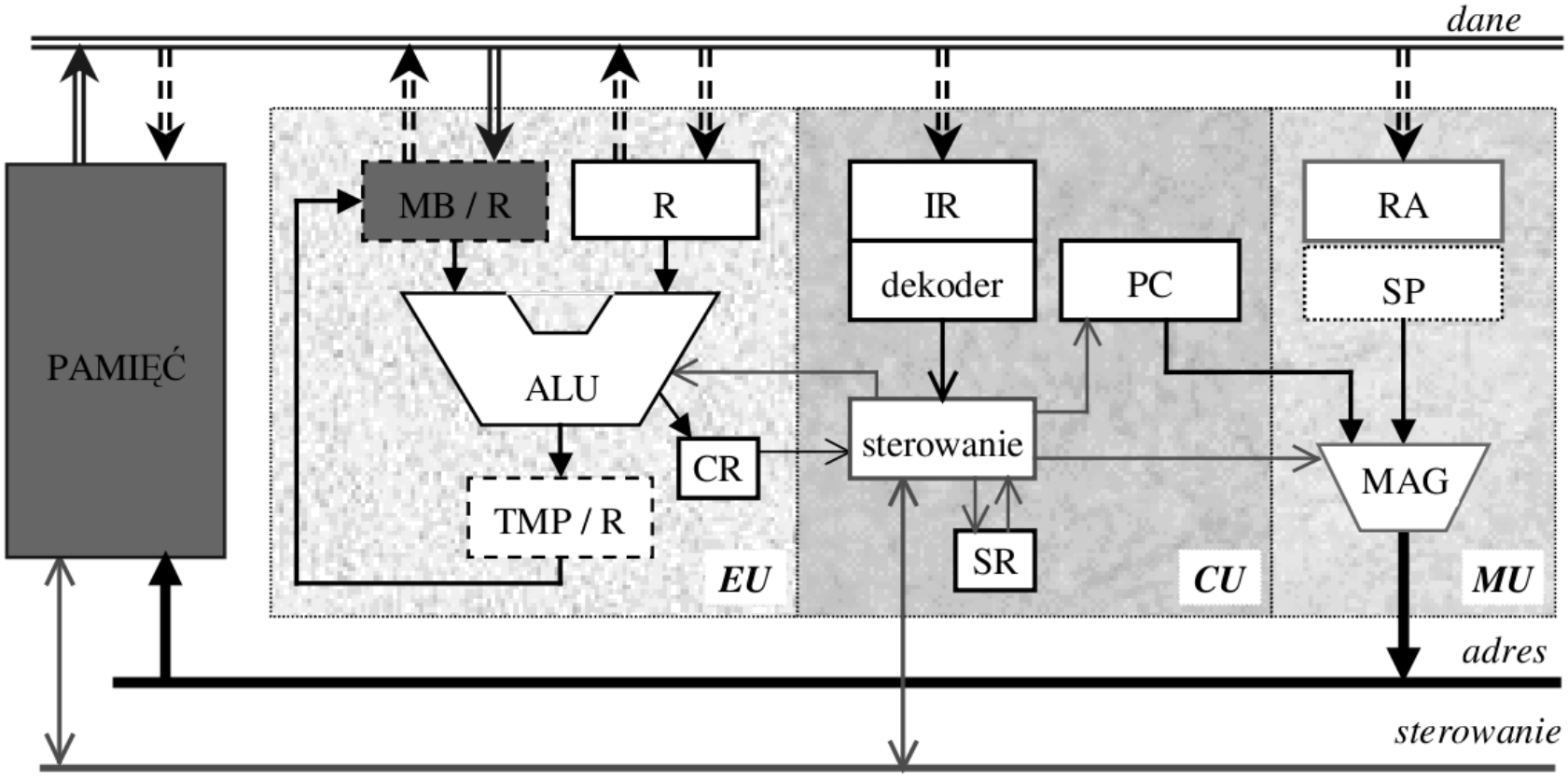
(R)ead



- *R* (odczyt danej): adres (RA) → pamięć → bufor (rejestr) MB

Architektura procesora sekwencyjnego (E)

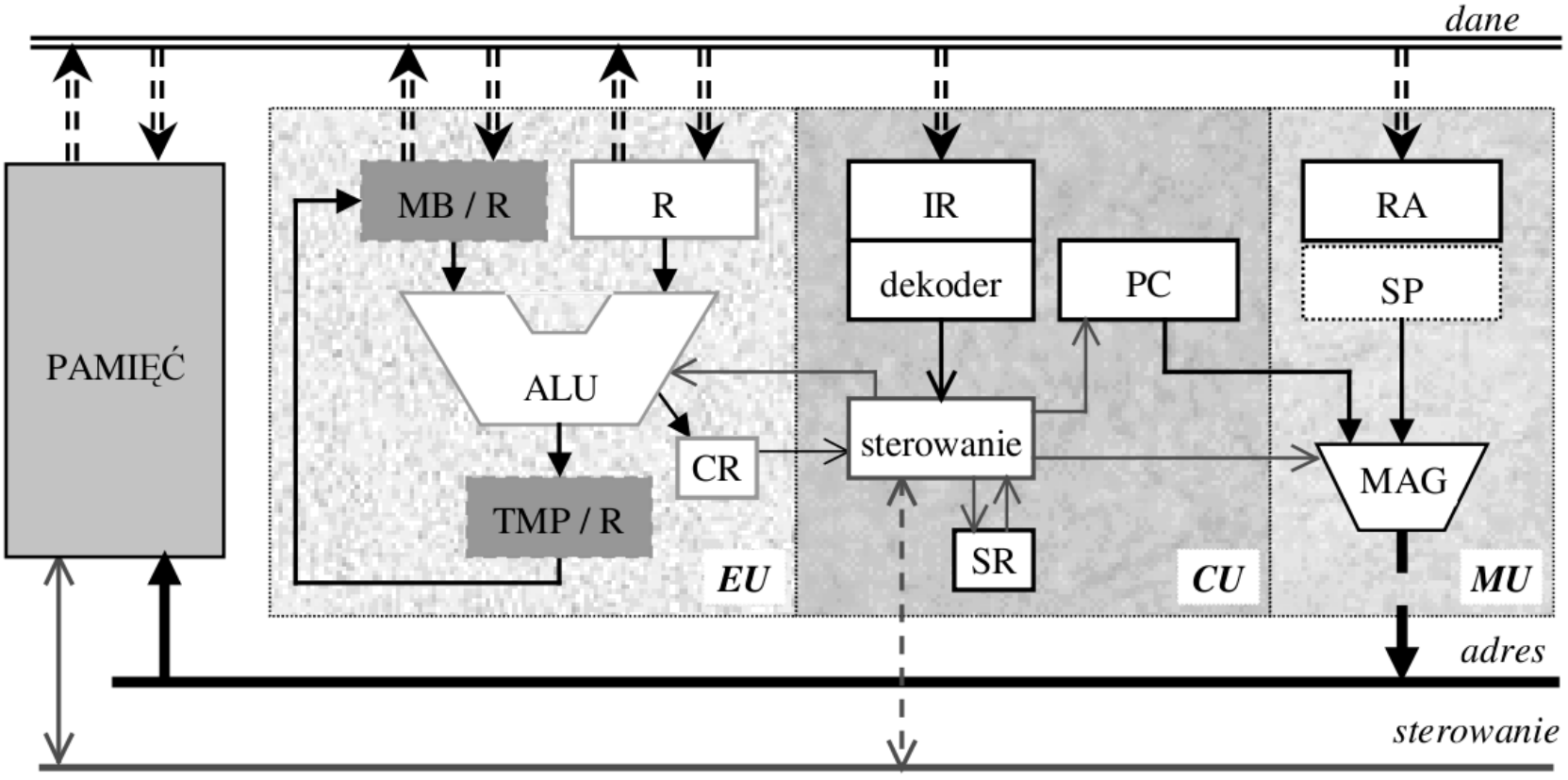
(E)xecute



- *E* (wytworzenie wyniku): ALU (MB/R, R) $\rightarrow$ bufor TMP/rejestr R

Architektura procesora sekwencyjnego (W)

(W)rite



- *W* (zapis wyniku): bufor MB $\rightarrow$ pamięć (adres (RA))

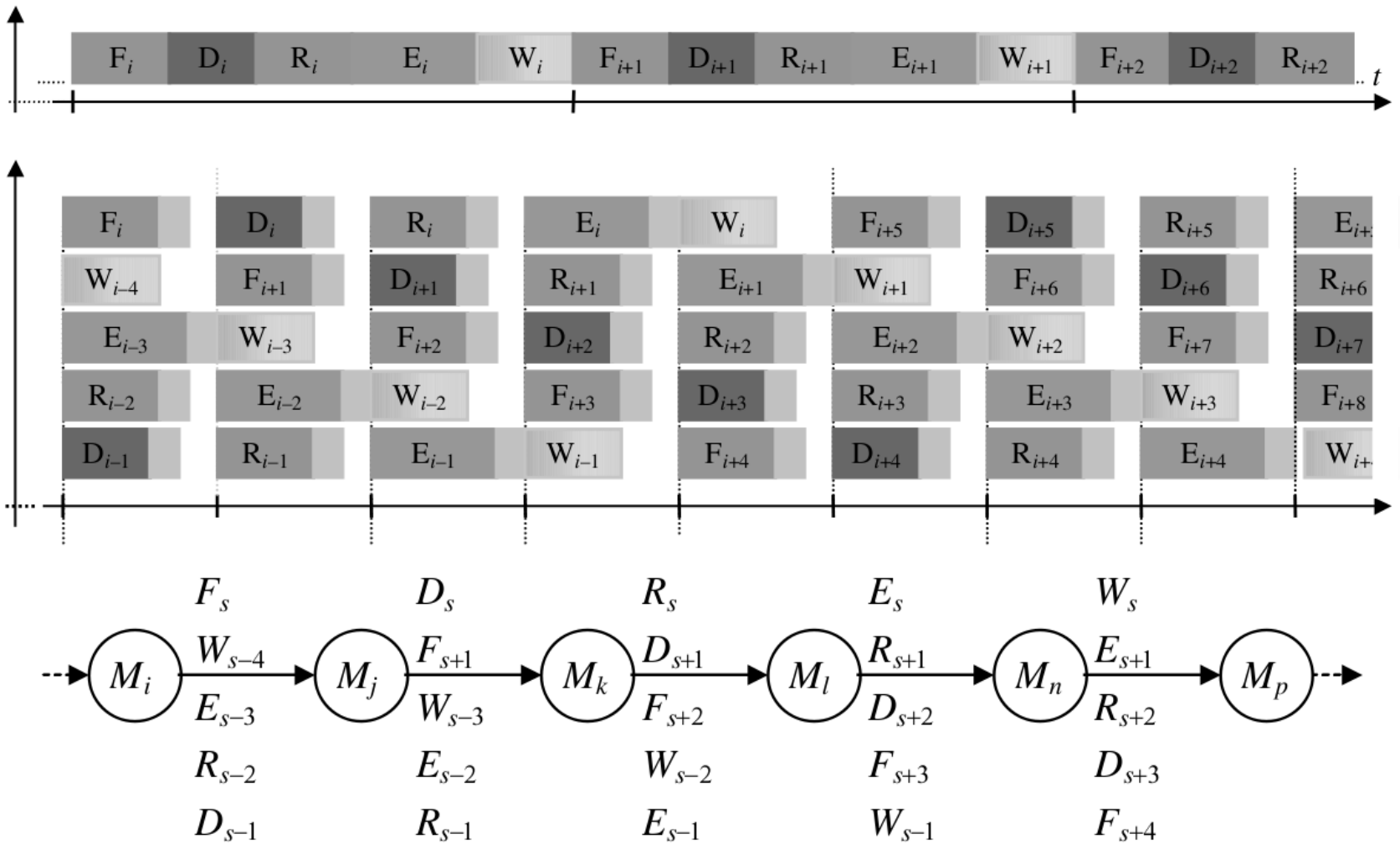
Koncepcja przetwarzania potokowego

- poszczególne etapy wykonują specjalizowane układy
- **jednokierunkowy przepływ danych** między układami
  - możliwe jednoczesne wykonanie różnych etapów
  - szybkość wykonania ogranicza najdłuższy etap i narzut separacji
    - Setup/hold time

czas wykonania etapu

- pobranie kodu rozkazu (fetch)
  - rozmiar kodu rozkazu
  - czas dostępu do pamięci (odczyt słowa lub jego części)
- dekodowanie (decode)
  - architektura listy rozkazów (złożoność i różnorodność działań)
  - struktura kodu rozkazu (niejednorodność)
  - pobranie operandu z pamięci (data *read*)
  - tryb adresowania i czas dostępu do pamięci (odczyt)
- wykonanie (execute)
  - złożoność wykonywanych działań
- zapis wyniku do rejestru (put *away*) (lub pamięci (data *write*)) (czas dostępu do pamięci (zapis słowa lub jego części))

Procesor potokowy



- Cykle procesora potokowego (w uproszczeniu)

Program

- Funkcje języków wysokiego poziomu
  - zwięzły opis algorytmu
  - ukrycie detali implementacyjnych
  - naturalny opis programowania strukturalnego i obiektowego
- Funkcje kodu maszynowego
  - implementacja operacji maszyny
- Przepaść semantyczna
  - Koncepcja CISC

Architektura klasyczna CISC

- CISC – Complex Instruction Set Computer
- Lista rozkazów
  - rozkazy realizujące działania proste i skomplikowane
  - rozbudowane sposoby (tryby) adresowania
  - argumenty umieszczone zwykle w pamięci
  - stałe w dodatkowych słowach kodu
  - niejednolita struktura – różna liczba słów kodu
- Organizacja – rozwiązania intuicyjne
  - akumulator lub niewiele rejestrów uniwersalnych
  - większość argumentów w pamięci, rejestry specjalizowane
  - trudne buforowanie i dekodowanie rozkazów (zmienny rozmiar)
- Skutki
  - zmienny czas wykonania tych samych etapów przetwarzania
  - bariera przepustowości pamięci

Analiza wykonania rozkazów w rozwiązaniach klasycznych (CISC)

analiza:

- rozbudowana lista rozkazów i nieregularna struktura kodu
  - zmienny czas pobrania kodu i dekodowania rozkazu
  - skomplikowany dekoder (układ sekwencyjny)
  - skomplikowane układy wykonawcze, zmienny czas wykonania działań
- większość operandów w pamięci
  - częste konflikty dostępu podczas wykonania etapów F, R i W
  - długi czas wykonania etapów F, R i W

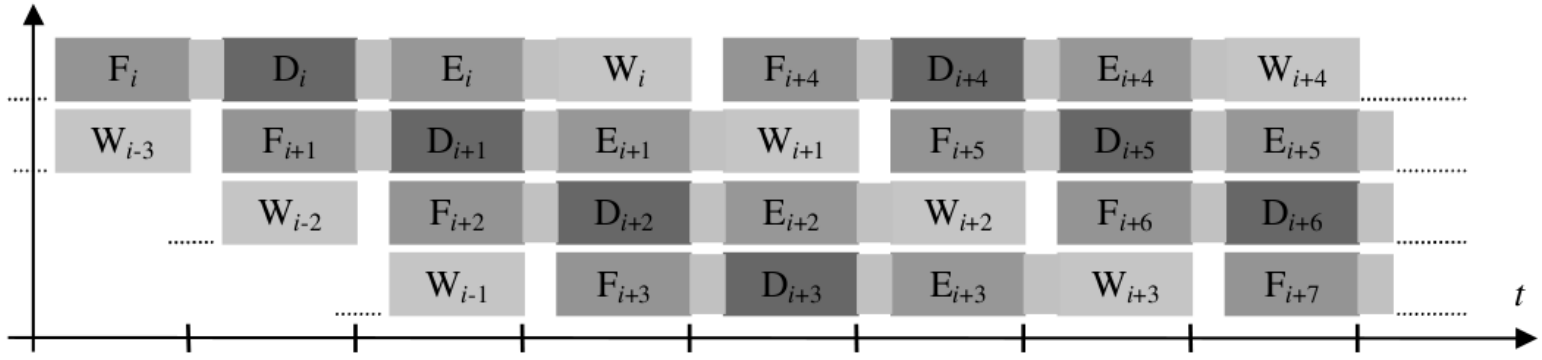
wnioski:

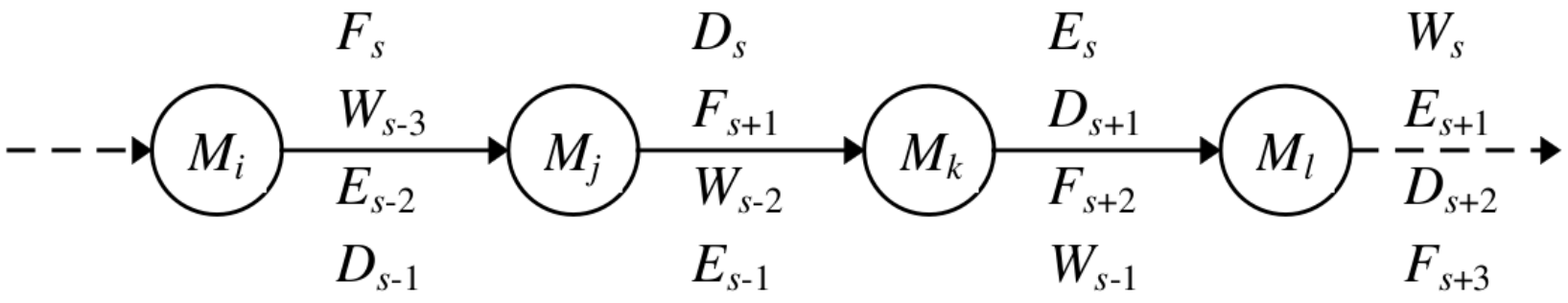
- uproszczona lista rozkazów oraz stały rozmiar i struktura słowa kodu:
  - stały czas pobrania kodu i dekodowania rozkazu
  - prosty dekoder kombinacyjny
  - proste układy wykonawcze, krótki czas wykonania podstawowych działań
- większość operandów w rejestrach procesora:
  - wyeliminowanie etapu R i krótszy czas etapu W (load/store)
  - rzadkie konflikty dostępu podczas wykonania etapów F, W

Koncepcja RISC

- RISC – Reduced Instruction Set Computer
  $\longrightarrow$!redukcja dotyczy różnorodności, a nie liczby instrukcji (racjonalizacja)!
- *Lista rozkazów*
  - rozkazy proste
  - proste tryby adresowania
  - specjalne rozkazy komunikacji z pamięcią
  - ograniczony zakres i krótkie kody stałych
  - jednolita struktura kodu
- *Organizacja* (zasada lokalności, buforowanie informacji)
  - dużo rejestrów uniwersalnych
  - buforowanie informacji (kolejka rozkazów, cache)
- *Korzyści*
  - prawie stały czas wykonania tych samych etapów przetwarzania
  - podobny czas wykonania różnych etapów przetwarzania
  - łatwa implementacja potoku
  - możliwe włączenie niektórych rozkazów CISC (o minimalnym wpływie na μarchitekturę)

Procesor potokowy RISC





Cykle procesora potokowego RISC

- Cykle procesora potokowego RISC

Potokowe wykonanie rozkazów w architekturze RISC

Uproszczona lista rozkazów oraz stały rozmiar i struktura słowa kodu,

- prosty dekoder kombinacyjny
- proste układy wykonawcze, krótki czas wykonania działań

Większość operandów w rejestrach procesora

- wyeliminowanie etapu R i krótszy czas etapu W (load/store)
- konflikty dostępu tylko podczas wykonania rozkazów load/store (E/W)

Niezbędne skrócenie czasu pobierania kodu z pamięci

- buforowanie pamięci

Nieuniknione przestoje (buble) wskutek konfliktów

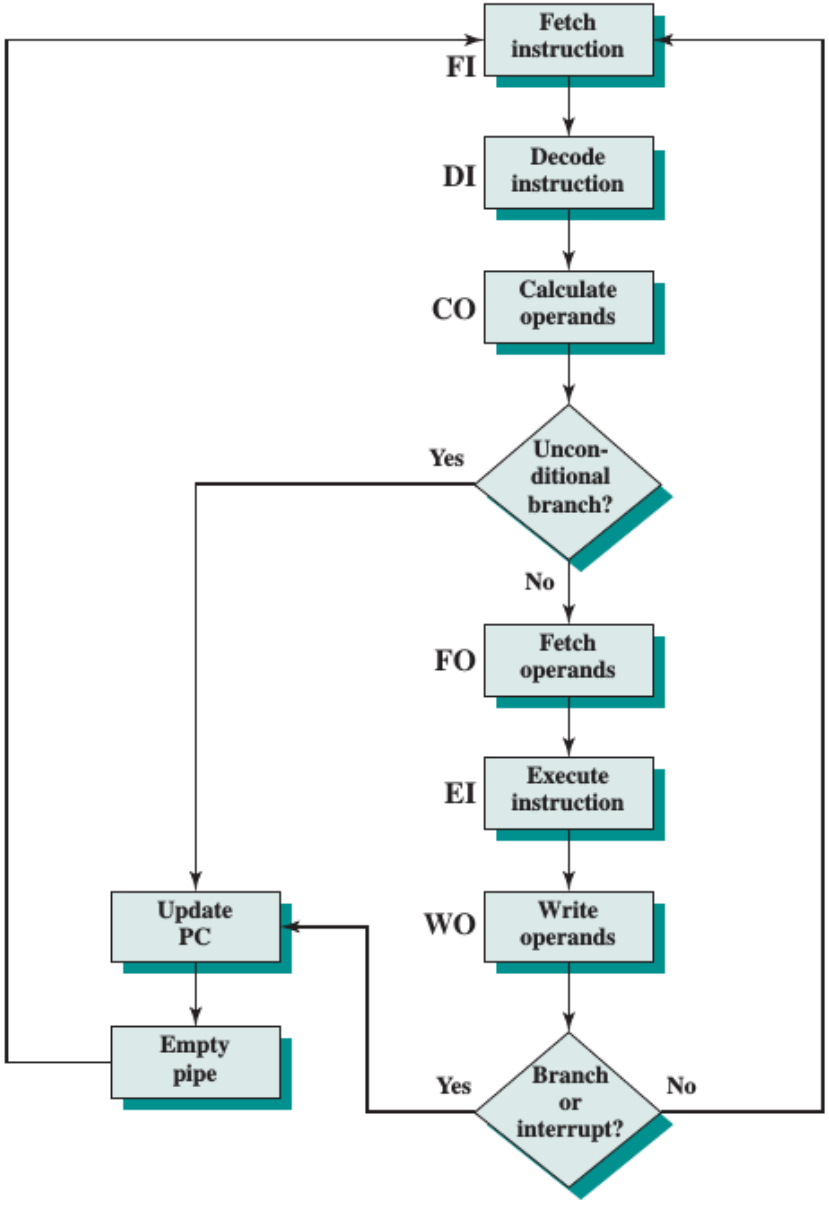
- spekulacyjne wykonanie niektórych rozkazów
- zmiana kolejności przetwarzania rozkazów

Przypuszczenie

- podział etapów F, D, E, W na podetapy
- wzrost przepustowości
- ...

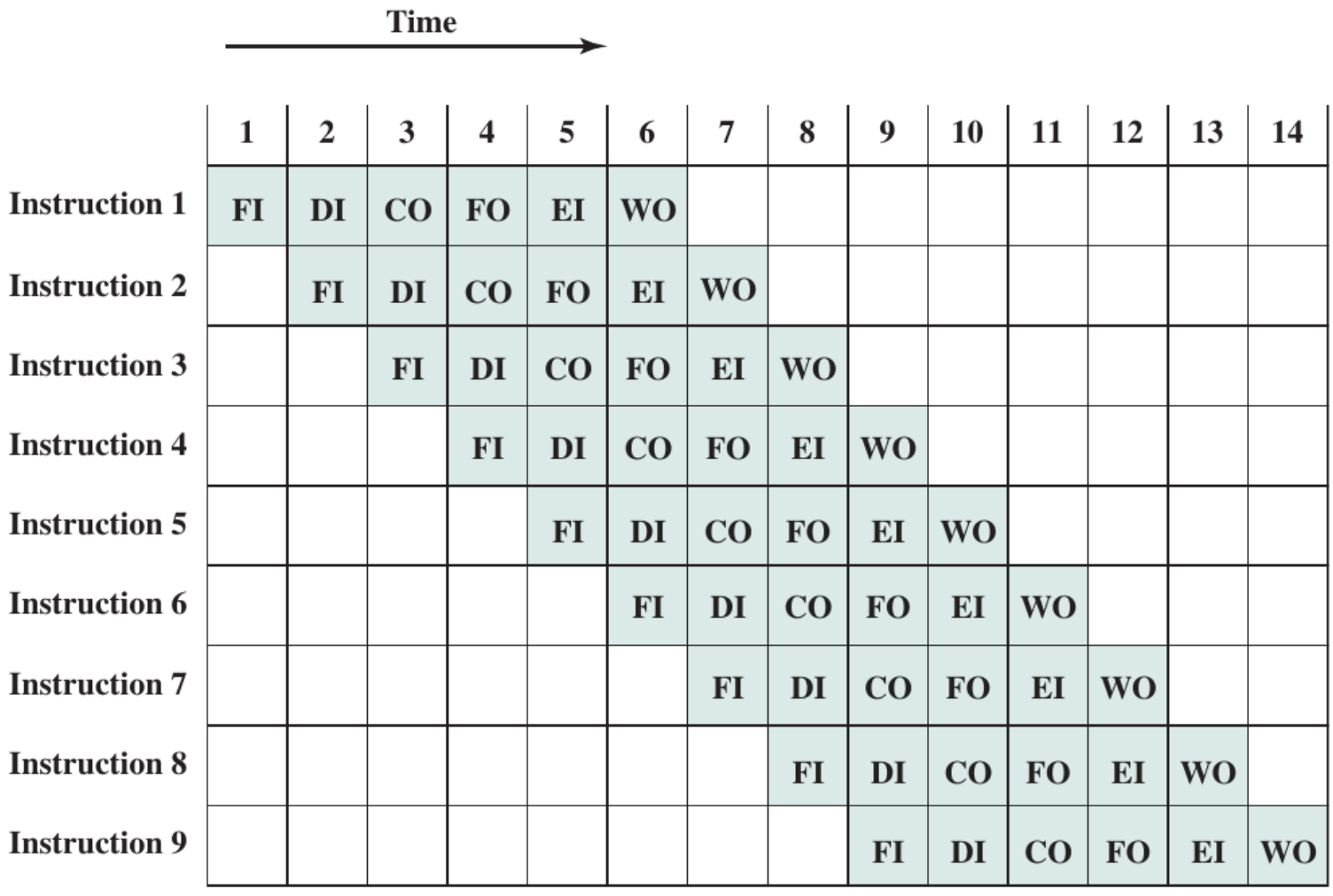
Przetwarzanie potokowe

Schemat działania



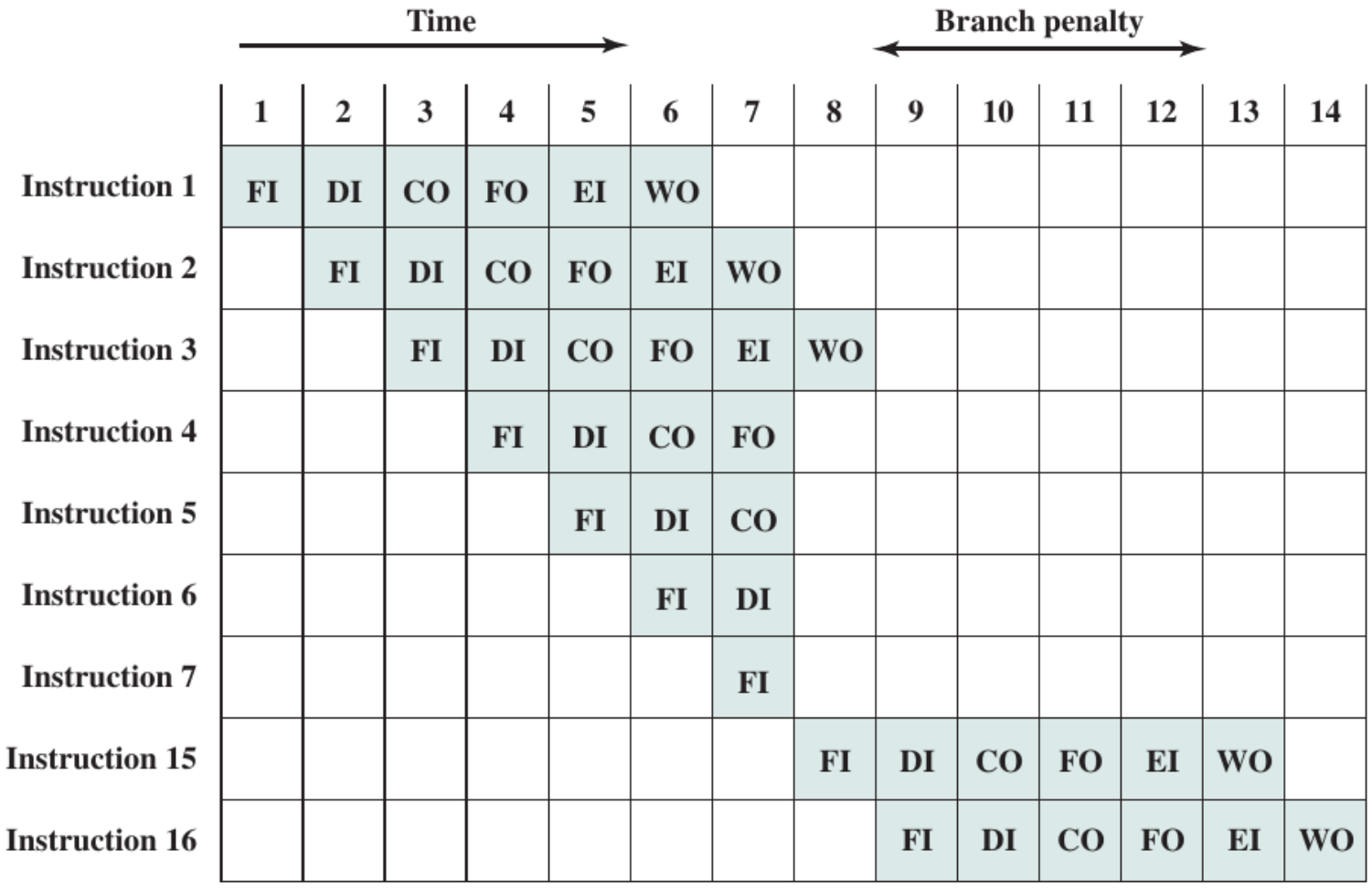
Przetwarzanie potokowe

Wykonanie rozkazów

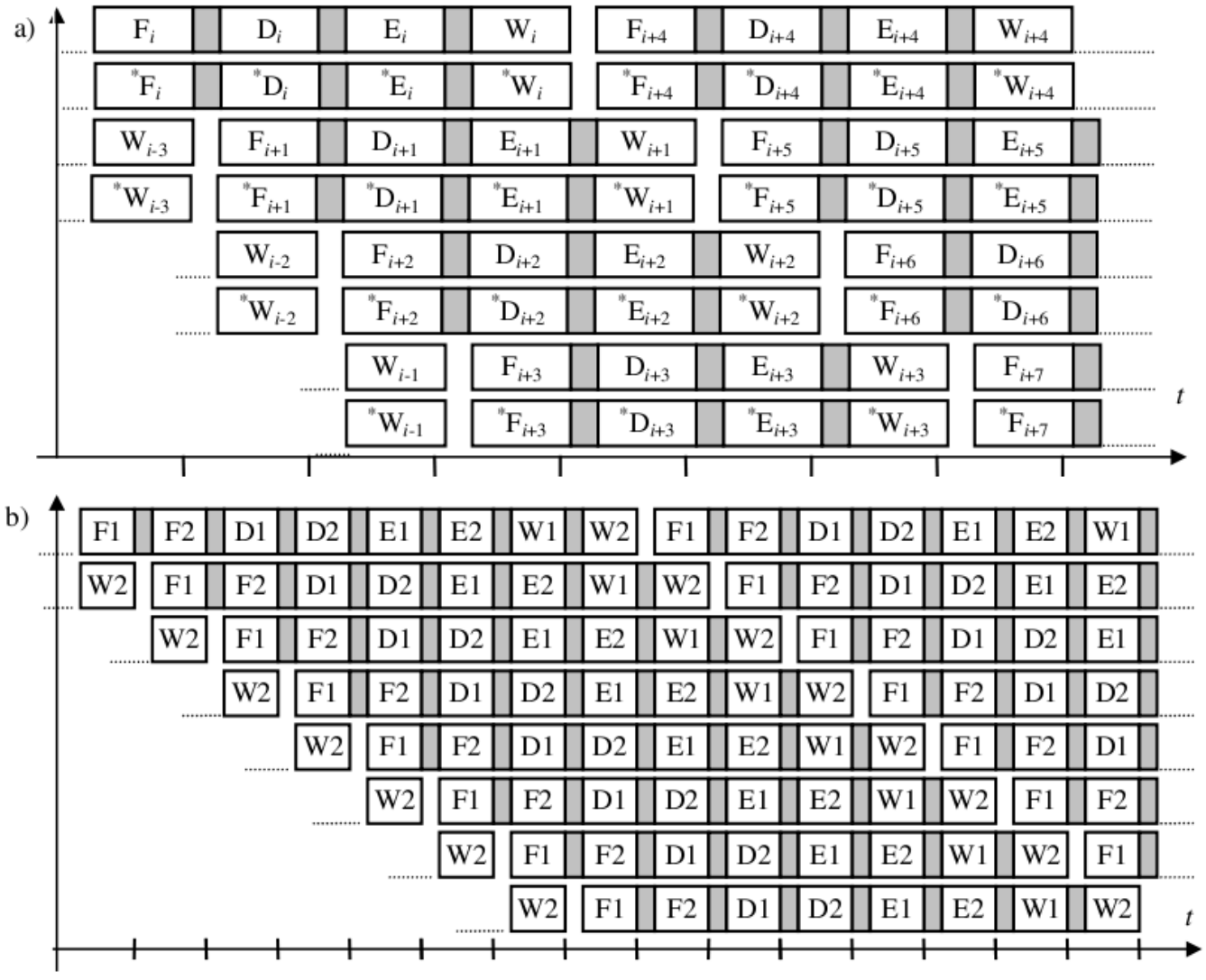


Przetwarzanie potokowe

Rozgałęzienie



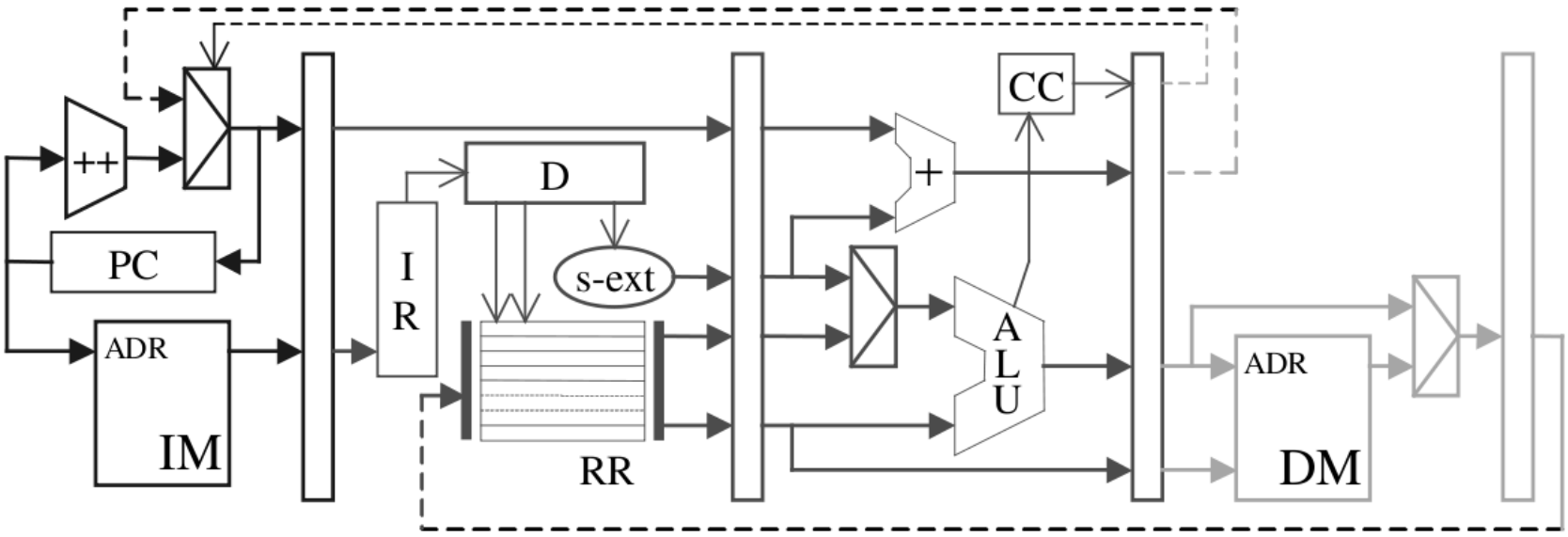
Przetwarzanie potokowe i superpotokowe



- Schematy przetwarzania: a) superskalarne, b) superpotokowe

Architektura procesora potokowego (rejestrowa)

$$\ldots \mathrm{RR}_{i+3} \rightarrow \mathrm{RR}_{i+2} \rightarrow \mathrm{RR}_{i+1} \rightarrow \mathrm{RR}_{i} \rightarrow \ldots$$



F : RR_{i+3} (IM – bufor pamięci podręcznej kodu, PC – licznik rozkazów)

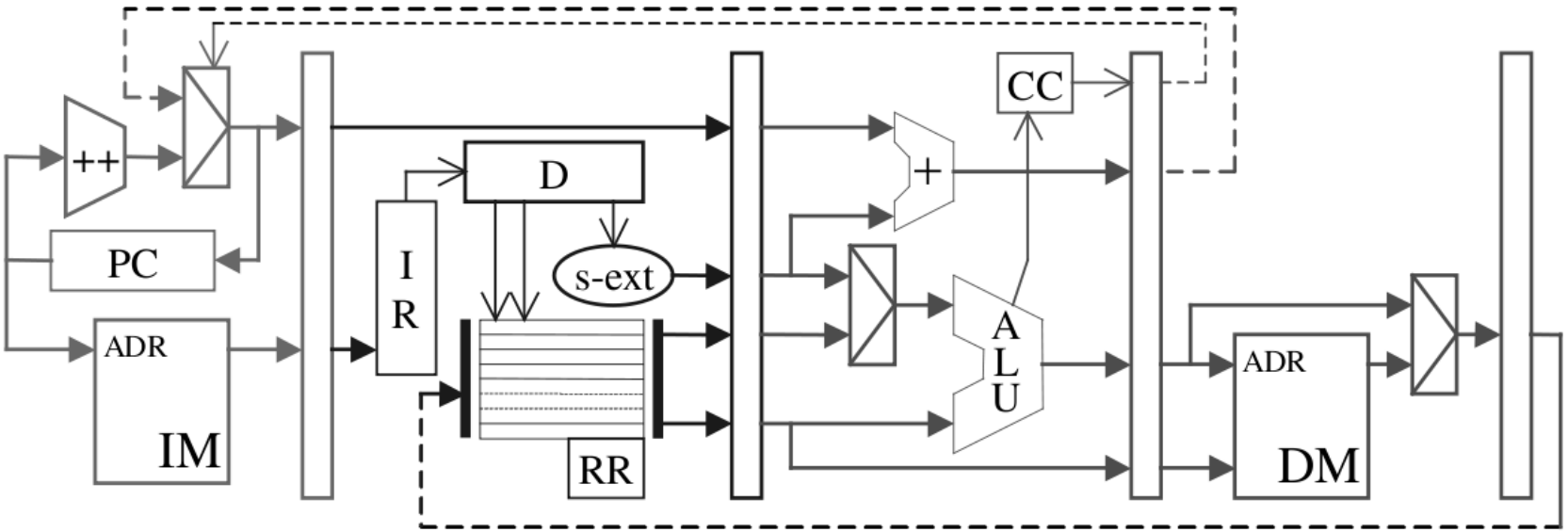
D : RR_{i+2} (IR – rejestr rozkazów, D – dekoder, RR – plik rejestrów)

E : RR_{i+1} (ALU – jednostka arytmetyczno-logiczna, CC – kod warunków)

W : RR_{i} (DM – bufor pamięci podręcznej danych (load/store))

Architektura procesora potokowego (rejestrowa)

$$\ldots \mathrm{RR}_{i+4} \rightarrow \mathrm{RR}_{i+3} \rightarrow \mathrm{RR}_{i+2} \rightarrow \mathrm{RR}_{i+1} \rightarrow \ldots$$



F : RR_{i+4} (IM – bufor pamięci podręcznej kodu, PC – licznik rozkazów)

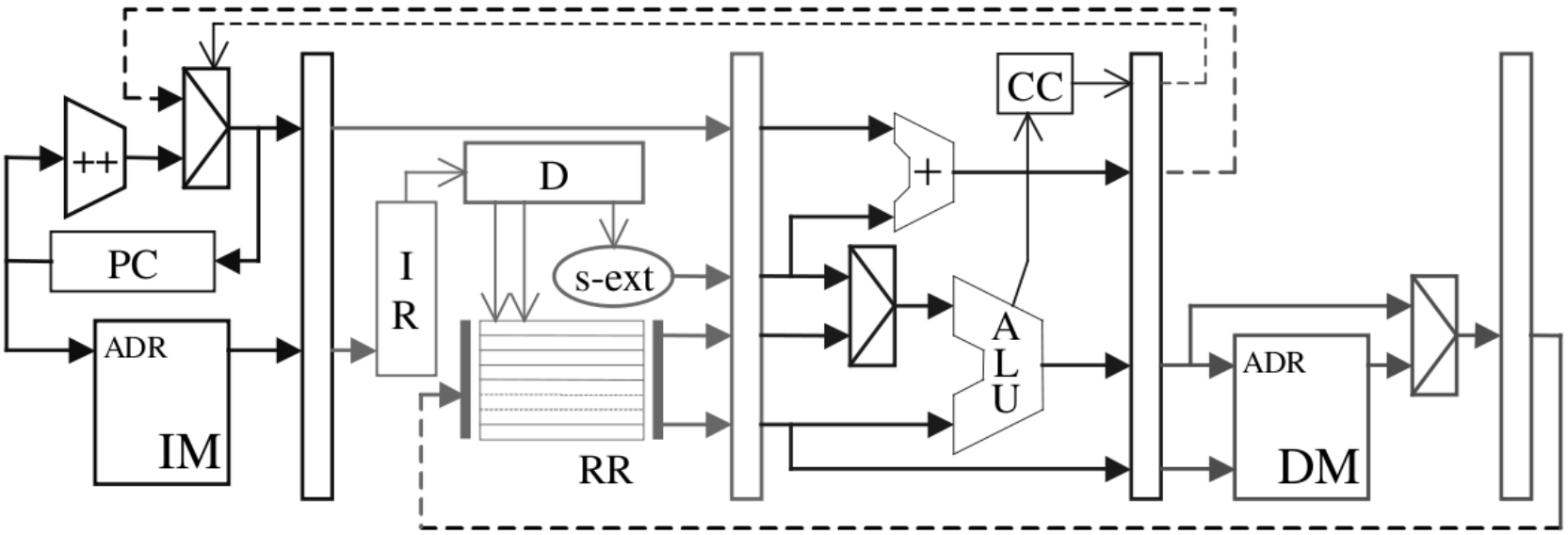
D : RR_{i+3} (IR – rejestr rozkazów, D – dekoder, RR – plik rejestrów)

E : RR_{i+2} (ALU – jednostka arytmetyczno-logiczna, CC – kod warunków)

W : RR_{i+1} (DM – bufor pamięci podręcznej danych (load/store))

Architektura procesora potokowego (rejestrowa)

$$\ldots \mathrm{RR}_{i+5} \rightarrow \mathrm{RR}_{i+4} \rightarrow \mathrm{RR}_{i+3} \rightarrow \mathrm{RR}_{i+2} \rightarrow \ldots$$



F : RR_{i+5} (IM – bufor pamięci podręcznej kodu, PC – licznik rozkazów)

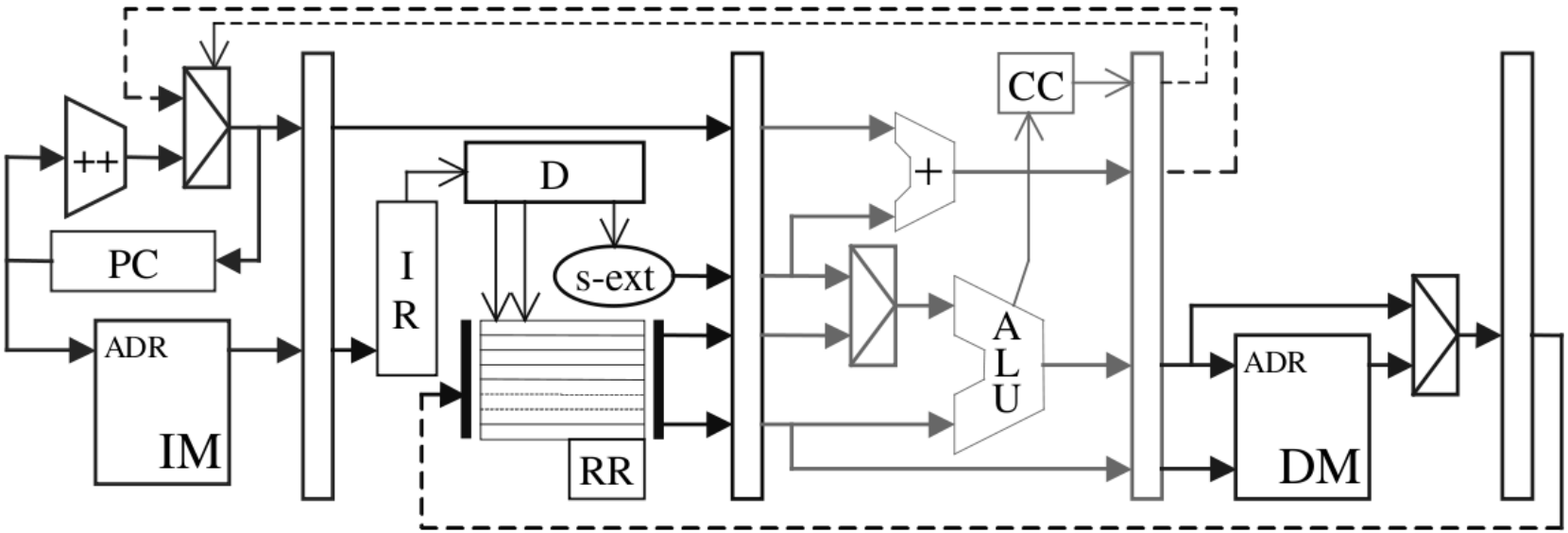
D : RR_{i+4} (IR – rejestr rozkazów, D – dekoder, RR – plik rejestrów)

E : RR_{i+3} (ALU – jednostka arytmetyczno-logiczna, CC – kod warunków)

W : RR_{i+2} (DM – bufor pamięci podręcznej danych (load/store))

Architektura procesora potokowego (rejestrowa)

$$\ldots \mathrm{RR}_{i+6} \rightarrow \mathrm{RR}_{i+5} \rightarrow \mathrm{RR}_{i+4} \rightarrow \mathrm{RR}_{i+3} \rightarrow \ldots$$



F : RR_{i+6} (IM – bufor pamięci podręcznej kodu, PC – licznik rozkazów)

D : RR_{i+5} (IR – rejestr rozkazów, D – dekoder, RR – plik rejestrów)

E : RR_{i+4} (ALU – jednostka arytmetyczno-logiczna, CC – kod warunków)

W : RR_{i+3} (DM – bufor pamięci podręcznej danych (load/store))

Przetwarzanie potokowe

Ograniczenia

- Bufory pośrednie i operacje na nich
- Złożoność zależności pomiędzy rejestrami i pamięcią
  - Układy śledzące zależności
- Opóźnienia rejestrów potoku (wysycenie potoku)