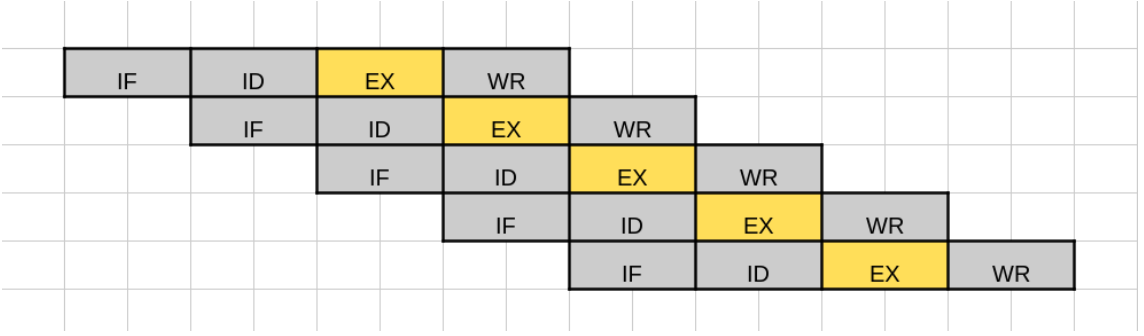
Przetwarzanie równoległe

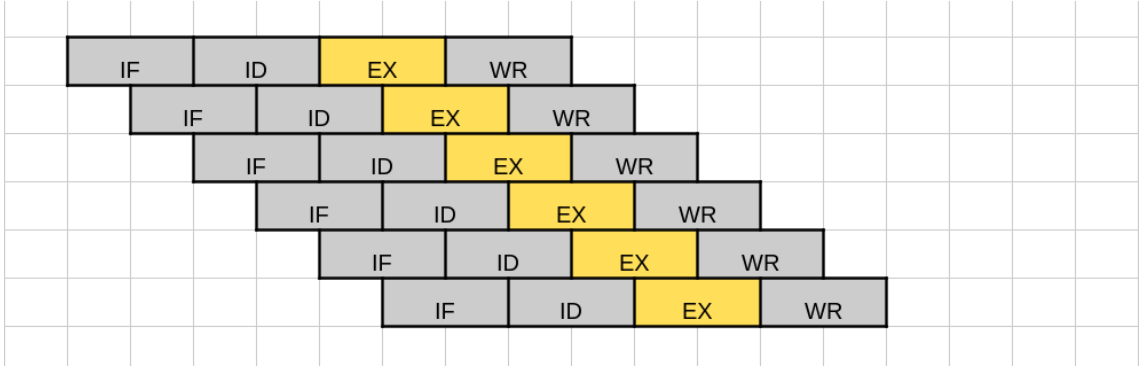
Potok

- Rozbicie cyklu rozkazu: IF, ID, (OF), EX, WR
- Wykonywanie operacji „na zakładkę”
- Zawsze tylko jeden rozkaz jest na etapie EX
- Optymalnie: wydanie 1 rozkazu na cykl potoku



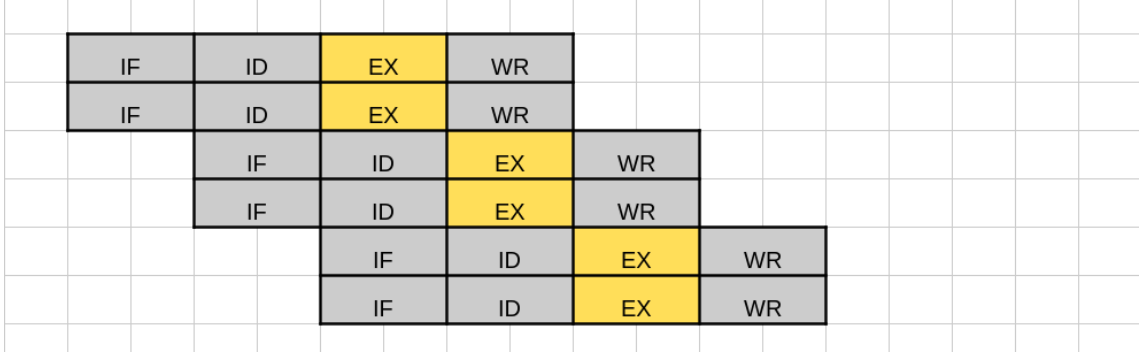
Superpotok

- Rozbicie etapów rozkazu na mniejsze
- Więcej rozkazów w tym samym czasie
- Większa równoległość w czasie
- Jednostki realizujące etapy muszą działać „na dwa fronty”
  - praca w dwóch pół-cyklach zegara potoku

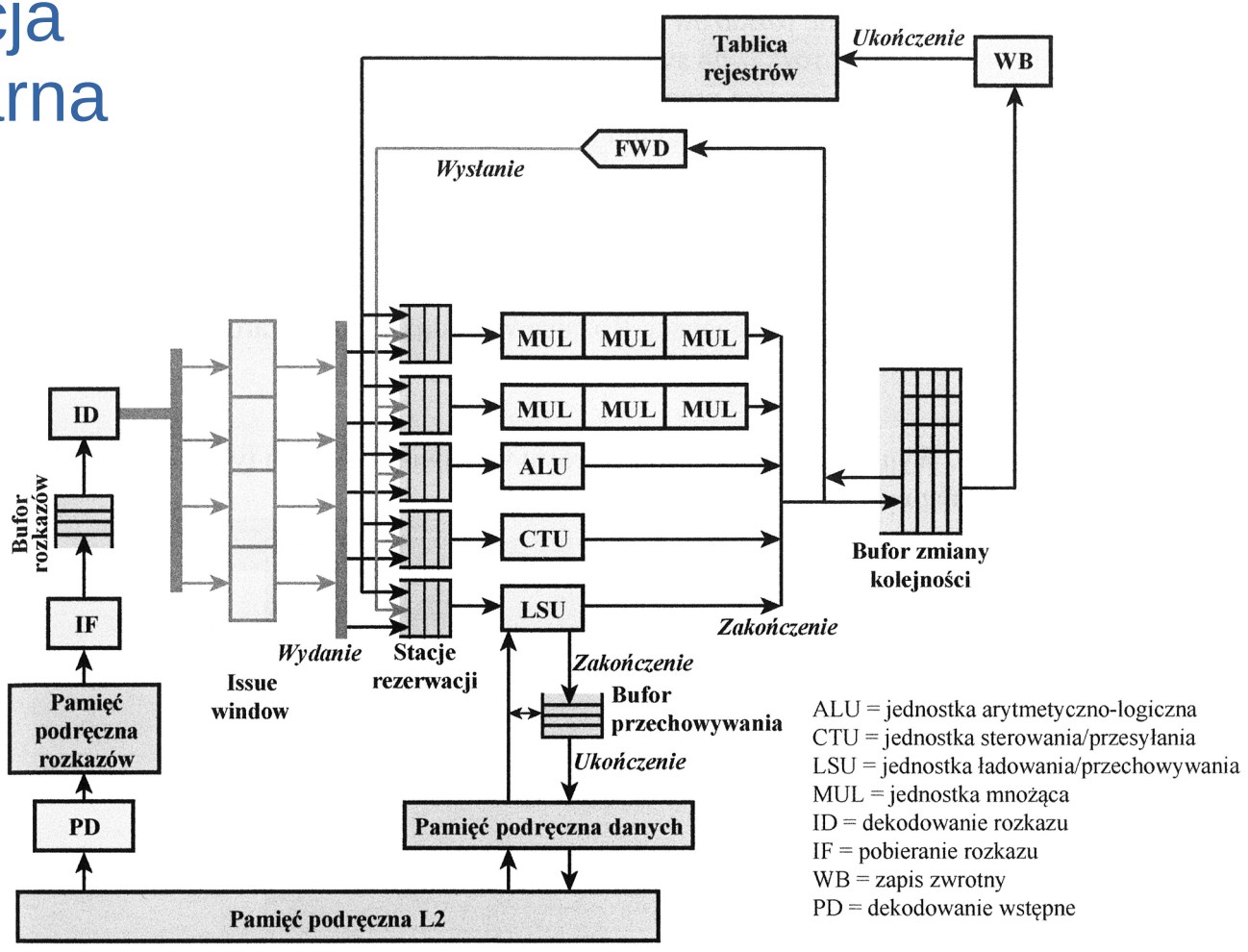


Superskalarność

- Uruchamiamy kilka potoków *równolegle*
- Jednocześnie wykonywane IF, ID, EX, …
  - zwielokrotnienie jednostek wykonawczych
- Rozwiązanie konfliktów między rozkazami



Organizacja superskalarna



Rysunek 18.2. Ogólna organizacja superskalarna (stopnia 4)

Źródło: Wykorzystano za zgodą profesora Rogera Kieckhafera z Michigan Technological University.

Konflikty zasobów (przypomnienie)

- (Prawdziwa) zależność danych (I1⇒I2, I3⇒I4)
- Zależność proceduralna
- Konflikt zasobów
  - jednoczesny dostęp do ALU, magistrali, rejestrów…
- Zależność od **wyjścia** (I1⇒I3)
- Antyzależność – zależność WAR – *write after read* (I2⇒I3)

- ```
  ADD EAX, ECX
  MOV EBX, EAX
  ```

- ```
  SUB EAX, ECX
  JNZ loop
  ```

- ```
  I1: R3 ← R3 op R5
  I2: R4 ← R3 + 1
  I3: R3 ← R5 + 1
  I4: R7 ← R3 op R4
  ```
  ```

Równoległość rozkazów i maszyny

```
Load R1, R2          Add R3, R3,'1'
Add  R3, R3,'1'      Add R4, R3, R2
Add  R4, R4, R2      Store [R4], R0
```

- Równoległość poziomu rozkazu:
  niezależność rozkazów w sekwencji programu
- Równoległość poziomu maszyny:
  zdolność do wykrycia i wykorzystania równ. rozkazu

Zapewnianie równ. poziomu maszyny

- Poszukiwanie (przyszłych) rozkazów, które można wykonać równolegle
  - kolejność pobierania rozkazów
  - kolejność wykonania rozkazów
  - kolejność zapisu wyników do rejestrów/pamięci
- Wydawanie rozkazów (*wysyłanie do wykonania*) w superskalarze
  - wydawanie w kolejności + wykonywanie w kolejności
  - wydawanie w kolejności + wykonywanie poza kolejnością
  - wydawanie poza kolejnością + wykonywanie poza kolejnością

CYKL	DEKOD.		WYKON.			ZAPIS	
1	I1	I2					
2	I3	I4	I1	I2			
3	I3	I4	I1				
4		I4			I3	I1	I2
5	I5	I6			I4		
6		I6		I5		I3	I4
7				I6			
8						I5	I6

	DEKOD.		WYKON.			ZAPIS	
1	I1	I2					
2	I3	I4	I1	I2			
3		I4	I1		I3	I2	
4	I5	I6			I4	I1	I3
5		I6		I5		I4	
6				I6		I5	
7						I6	

	DEKOD.		OKNO	WYKON.			ZAPIS	
1	I1	I2						
2	I3	I4	I1,I2	I1	I2			
3	I5	I6	I3,I4	I1		I3	I2	
4			I4,I5,I6		I6	I4	I1	I3
5			I5		I5		I4	I6
6							I5	

I1 potrzebuje 2 cykle
I3 i **I4** – konflikt jednostki funkcjonalnej
I5 zależność od danych od I4
I5 i **I6** – konflikt jednostki funkcjonalnej

Ukończenie poza kolejnością

- Potrzebne układy do przywracania kolejności zapisu
- Problem z przerwaniami i wyjątkami:

```
ADD, R5, R5, R2
DIV R4, R0          // R0 = 0: wyjątek!
```

  - jeśli ADD jest wykonywane równolegle…

Wydawanie poza kolejnością

- Przy konflikcie dekodowanie jest wstrzymane
  - nie ma jak szukać przyszłych „niezależnych” rozkazów
- Używamy bufora (okno) – zdekodowane „na zaś”
- Wydanie rozkazu z okna, jeśli:

  1) rozkaz czeka na jedn. funkcjonalną, która jest wolna

  2)nie ma zależności/konfliktów dla tego rozkazu

Przemianowanie rejestrów

- Wykonanie/zapis poza kolejnością mogą prowadzić do zależności WAW i WAR
  - w danej chwili zawartość r-rów nie musi odpowiadać tej wynikającej z przebiegu programu
- Antyzależność i zależność od wyjścia
  - dostęp do tych samych rejestrów
  - optymalizacja kompilatora (maksymalizacja użycia r-rów)
- Przydzielanie rejestrów dynamicznie przy tworzeniu nowej wartości dla danego logicznego r-ru
- Aktualna wartość r-ru logicznego: ostatnio przydzielony alias

Przemianowanie rejestrów

```
R3 ← R3 op R5
R4 ← R3  + 1
R3 ← R5  + 1
R7 ← R3 op R4
```

$R3_b \leftarrow R3_a$ op $R5_a$
$R4_b \leftarrow R3_b + 1$
$R3_c \leftarrow R5_a + 1$
$R7_b \leftarrow R3_c$ op $R4_b$

ZASOBY

PLANOWANIE
WYKONANIE

Wielowątkowość

Idea: podział strumienia rozkazów na mniejsze (wątki), które można wykonać równolegle

- **Proces** – uruchomiony program
  - własne zasoby (przestrzeń pamięci wirtualnej) na stos, dane, program, atrybuty
  - wykonywany jako ścieżka (ślad) sekwencji programów, planowany przez OS: gotowy, uruchomiony, czekający...
- Przełączanie procesu
  - załadowanie innego procesu, zapisanie danych sterujących aktualnego proces (PSW, r-ry, SP, PC, etc.) i załadowanie danych nowego procesu
- **Wątek** – gotowa do uruchomienia jednostka pracy w procesie
  - własny PC, SP; wykonywany sekwencyjnie i przerywany na rzecz innego wątku.
- Przełączanie wątków
  - wymiana aktualnie realizowanego wątku (z przełączeniem danych sterujących – **mniejsze**!)

Wątki wielowątkowości

- W. z przeplotem (w. drobnoziarnista):
  - obsługa kilku kontekstów wątków
  - cykliczne przełączanie, pomijanie zablokowanych
- W. zablokowana (w. gruboziarnista):
  - wątek wykonywany, aż wywoła opóźnienie
- W. jednoczesna (SMT – simultaneous multithreading)
  - rozkazy z wielu wątków wykonywane równolegle na wielu jendostkach wykonawczych
- Ukł. wieloprocesorowe
  - każdy rdzeń zajmuje się własnym wątkiem

Multithreaded Processors

THEO UNGERER[1], BORUT ROBIČ[2] AND JURIJ ŠILC[3]

[1] *University of Augsburg, Department of Computer Science, Augsburg, Germany*
[2] *University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia*
[3] *Jožef Stefan Institute, Computer Systems Department, Ljubljana, Slovenia*
Email: Theo.Ungerer@informatik.uni-augsburg.de