CPU

Central Process Unit

étude de la structure de la CPU

exemple de processeur RISC

plan

- Organisation de la CPU
  - Organisation du processeur
  - Organisation des registres
- Cycle de l'instruction
  - Cycle normal
  - Interruptions

plan

- Reduced Instruction Set Computers
  - Caractéristiques
  - Exemple de machine RISC
    - Jeu d'instructions et registres
    - Cycle d'instruction
    - Réalisation

plan

- Pipeline
  - Pipeline de base
    - Pipeline à 3 étages
    - Pipeline à 5 étages
  - Les aléas
    - Aléa structurel
    - Aléa de données
    - Aléa de contrôle

plan

- Améliorations
  - Architecture superscalaire
  - Architecture VLIW

organisation

- unités de calcul (ALU, FPU)
- unité de commande
- registres (données, adresses, instruction, contrôle)
- bus interne

organisation

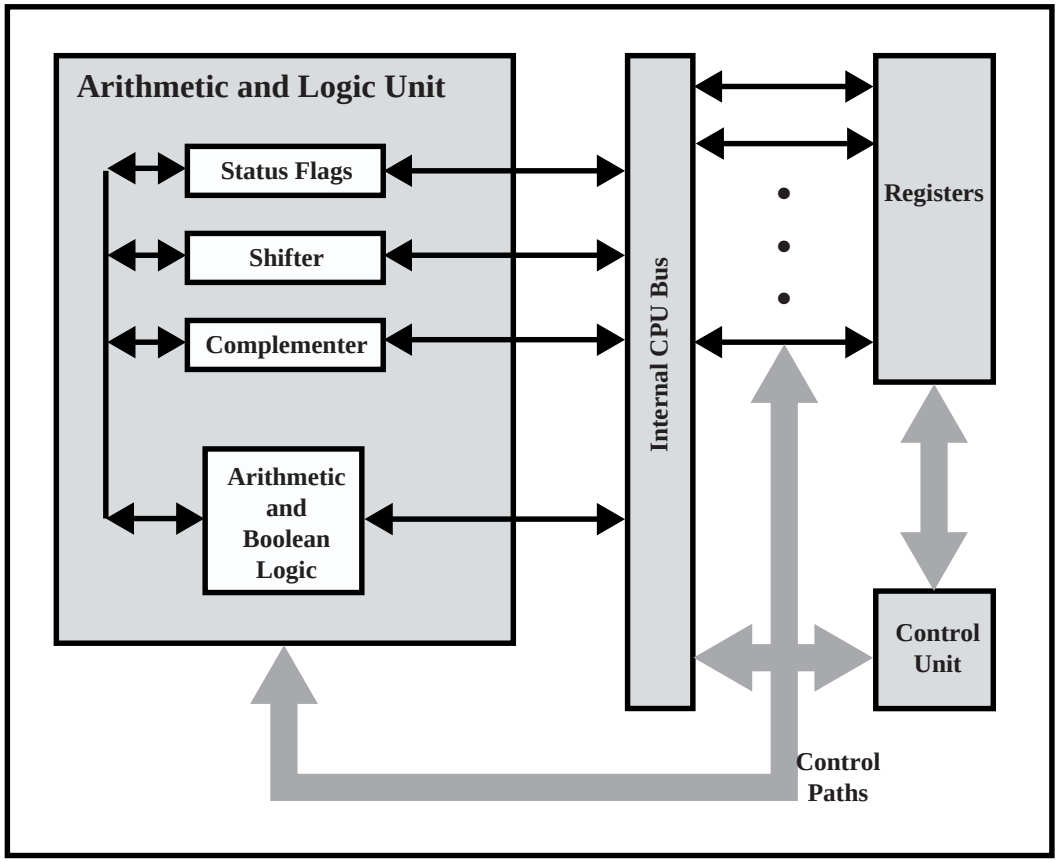


Figure 11.2 Internal Structure of the CPU

registres

mémoire interne à la CPU

- registres visibles par le programmeur
- registres de contrôle et de statuts
  - utilisés par la CPU
  - utilisés par le système

registres visibles

- générique : pas de restriction de contenu
- données
- adresses : souvent dévolus à un mode d'adressage
  - adresse de base de segment (e.g., pentium)
  - index
  - pointeur de pile en mémoire

registres visibles

– conditions (flags) :

suite de bits indépendants

positionnés en fonction du résultat d'une opération

lecture seule

choix de conception

– quelle proportion de registres spécialisés adopter ?
– quel nombre de registres adopter ?
– quelle taille de registres adopter ?

registre de contrôle et statuts

échange avec la mémoire principale

- compteur ordinal (PC)
  - adresse de la prochaine instruction à exécuter
- registre d'instruction (IR)
  - instruction en cours d'exécution

registre de contrôle et statuts

- registre d'adresse mémoire (MAR)
  - contient une adresse mémoire
  - directement connecté au bus d'adresse
- registre tampon mémoire (MBR)
  - contient un mot de données
  - directement connecté au bus de données
- des registres intermédiaires

registre PSW (Program Status Word)

inclut les booléens suivants :

– bit de signe du résultat
– le résultat est 0
– l'opération a généré une retenue
– le résultat d'une comparaison est une égalité
– une opération a provoqué un débordement
– le fonctionnement normal peut être interrompu
– mode privilégié
– ...

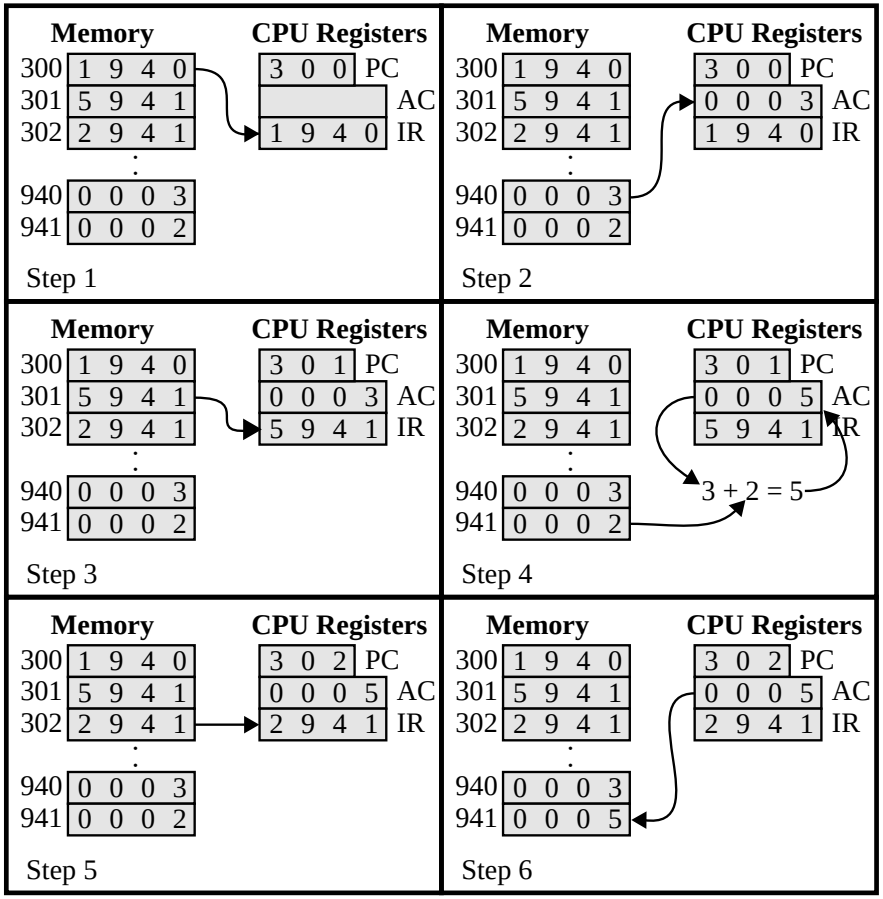


Figure 3.5 Example of Program Execution

cycle de l'instruction

- recherche de l'instruction (fetch)
  - lecture de l'instruction depuis la mémoire
- interprétation de l'instruction (decode)
  - détermination de l'opération correspondant à l'instruction
  - recherche des données (fetch data)
    - lecture de données dans la mémoire ou depuis un module d'E/S

cycle de l'instruction

- exécution (execute)
  - traitement des données (process data)
    - opérations arithmétiques ou logiques
  - écriture des données (write data)
    - écriture du résultat dans la mémoire ou vers un module d'E/S
- interruption (interrupt) : vérifie si une interruption est apparue

flot de données

- fetch :
  - $MAR \leftarrow PC$
  - l'unité de contrôle demande une lecture de la mémoire principale
  - le résultat de la lecture est placé dans MBR
  - $IR \leftarrow MBR$
  - $PC \leftarrow PC + 1$

flot de données

- exemple de cycle indirect pour decode
  - $MAR \leftarrow$ les N bits de poids faibles de MBR
  - l'unité de contrôle demande une lecture de la mémoire principale
  - le résultat de la lecture est placé dans MBR
  - $MAR \leftarrow MBR$
  - l'unité de contrôle demande une lecture de la mémoire principale
  - le résultat de la lecture est placé dans MBR

prévisibilité

les cycles fetch et decode sont simples et prévisibles

le cycle execute est imprévisible

interruptions

mécanisme selon lequel un module (E/S, mémoire) interrompt un cycle normal

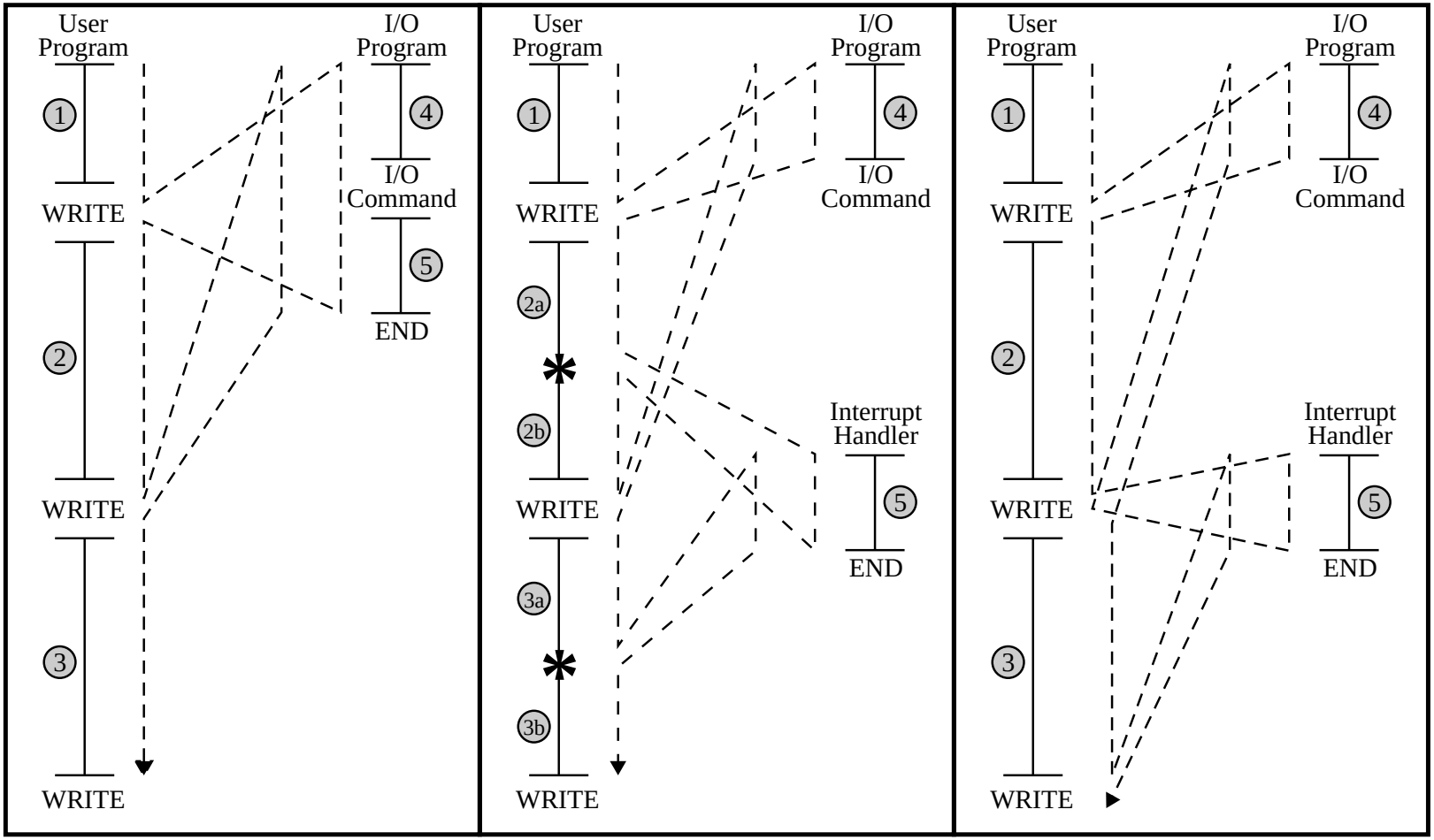
– programme : division par 0, ...
– E/S : fin d'une opération, erreur, ...
– problème matériel

permet au processeur de ne pas être dépendant des périphériques

interruptions

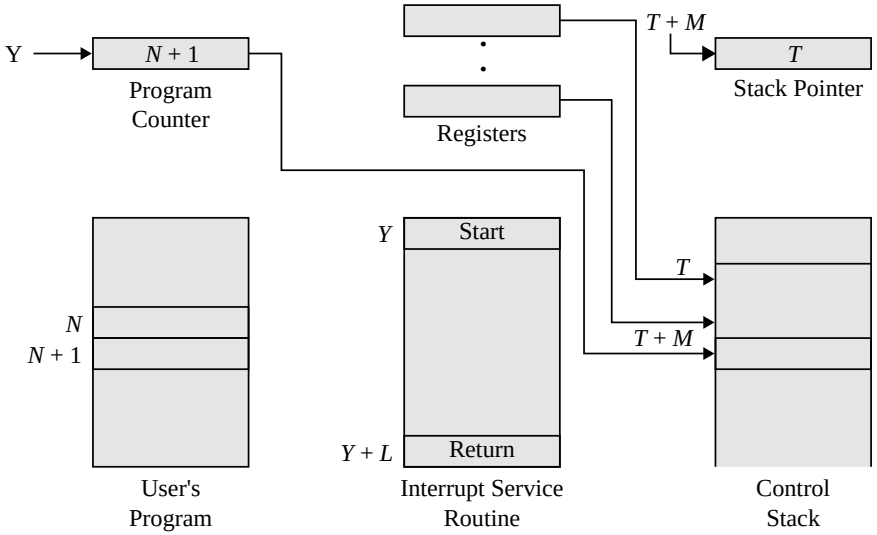
en cas d'interruption

- sauvegarde de l'état courant
  - PC
  - PSW
  - ...
- exécution d'une suite d'instructions traitant l'interruption (routine d'interruption)
- retour à l'état sauvegardé

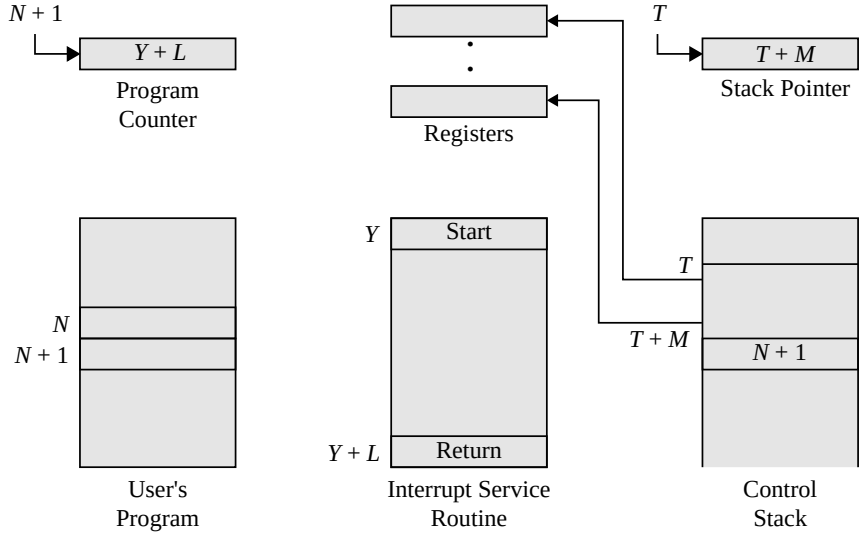


(a) No interrupts (b) Interrupts; short I/O wait (c) Interrupts; long I/O wait

Figure 3.7 Program Flow of Control Without and With Interrupts



(a) Interrupt occurs after instruction at location *N*



(b) Return from interrupt

Figure 6.8 Changes in Memory and Registers for an Interrupt

cycle d'interruption

simple et prévisible

- $MBR \leftarrow PC$
- $MAR \leftarrow$ une adresse mémoire ou sauvegarder PC
- l'unité de contrôle demande une écriture dans la mémoire principale
- $PC \leftarrow$ adresse de la routine d'interruption

le nouveau cycle commence par le fetch de la première instruction de la routine d'interruption

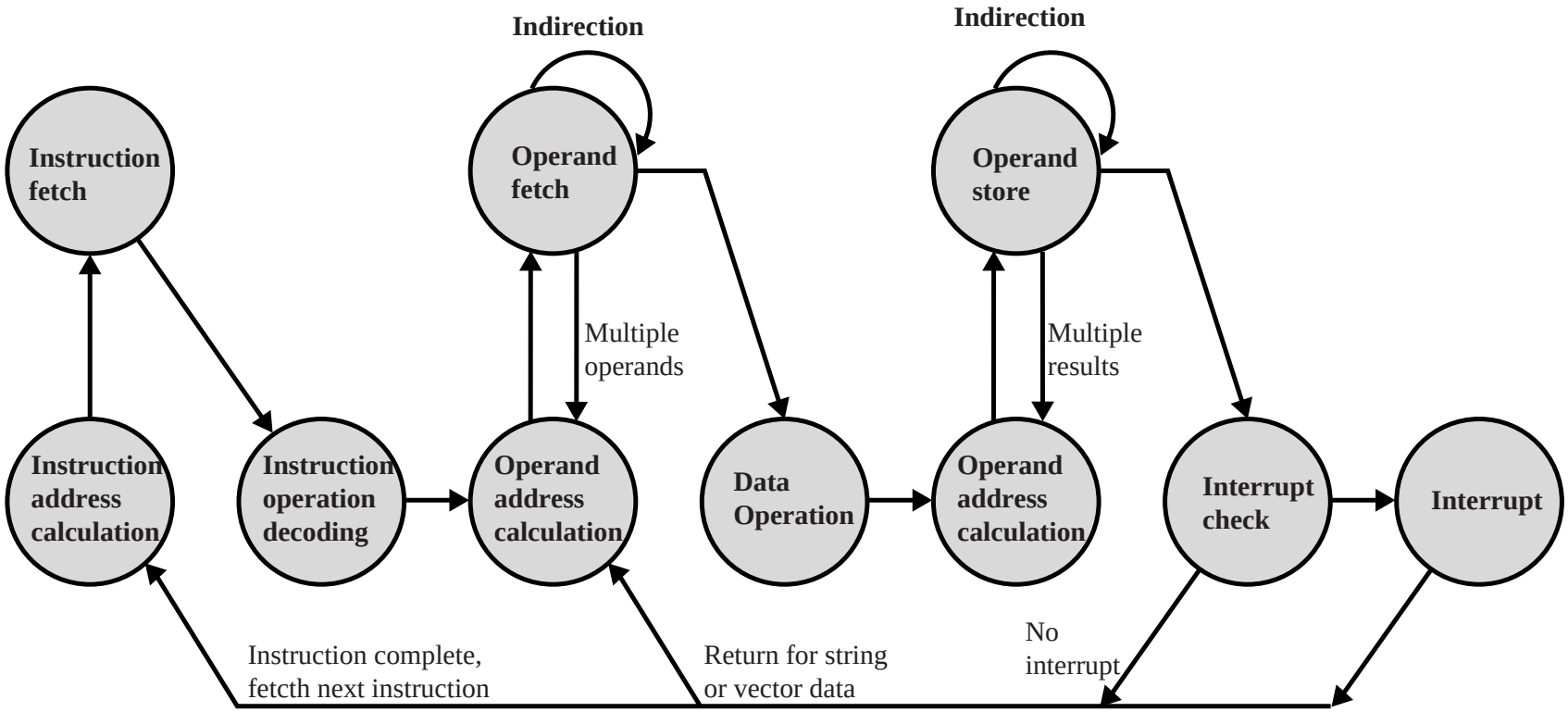


Figure 11.6 Instruction Cycle State Diagram

RISC

Reduced Instruction Set Computers

- les compilateurs produisent surtout des instructions simples
- les instructions complexes des CISC sont rarement exploitées
- ces dernières peuvent être recodées par des instructions simples

RISC

caractéristiques communes aux architectures RISC

– vers une instruction par cycle machine
– nombre de registres génériques important
– opérations registre-registre

RISC

– achitecture de type chargement-rangement
– modes d'adressage simples et peu nombreux
– formats d'instructions simples et peu nombreux
– taille d'instruction fixe, possédant des champs fixes

instructions simples donc unité de contrôle simple

exemple

machine utilisant 3 formats d'instructions

code 6 bits	Rs1 5 bits	Rd 5 bits	immédiat 16 bits

opcode 6 bits	Rs1 5 bits	Rs2 5 bits	Rd 5 bits	fonction 11 bits

opcode 6 bits	déplacement 26 bits

instructions

4 groupes d'instructions

– instructions de chargement-rangement (format 1)
– opérations registre-registre (format 2)
– opérations registre-immédiat (format 1)
– branchements (format 1)

format 3 non considéré

mémoires

- cache de données adressable à l'octet
- cache d'instructions adressable à l'octet
- RI : registre d'instruction
- CP : compteur ordinal
- NCP : adresse de l'instruction suivante

mémoires

- A, B, IMM entrées d'ALU
- SALU : sortie ALU
- COND : registre de condition/statut
- DMC : sortie mémoire de données
- des registres utilisateurs

cycle d'instruction

- **fetch (LI)**
  - RI $\leftarrow$ mémoire d'instruction(PC)
  - NCP $\leftarrow$ CP $+$ 4
- **decode (DI)**
  - A $\leftarrow$ registre($RI_{25..21}$)
  - B $\leftarrow$ registre($RI_{20..16}$)
  - IMM $\leftarrow$ $RI_{15..0}$

cycle d'instruction

- **execute/calcul de l'adresse effective (EX)** selon l'opcode, une des 4 opérations suivantes est réalisée :
  - accès mémoire
    - SALU $\leftarrow$ A + IMM
  - opération registre-registre
    - SALU $\leftarrow$ A op B

cycle d'instruction

- opération registre-immédiat
  - SALU $\leftarrow$ A op IMM
- branchement
  - SALU $\leftarrow$ NCP $+$ IMM
  - COND $\leftarrow$ A op 0

cycle d'instruction

- **accès mémoire/branchement (MEM)**
  - chargement
    - DMC ← mémoire données(SALU)
    - CP ← NCP
  - rangement
    - mémoire données(SALU) ← B
    - CP ← NCP

cycle d'instruction

- branchement
  - si COND alors CP $\leftarrow$ SALU sinon CP $\leftarrow$ NCP
- autres instructions
  - CP $\leftarrow$ NCP

cycle d'instruction

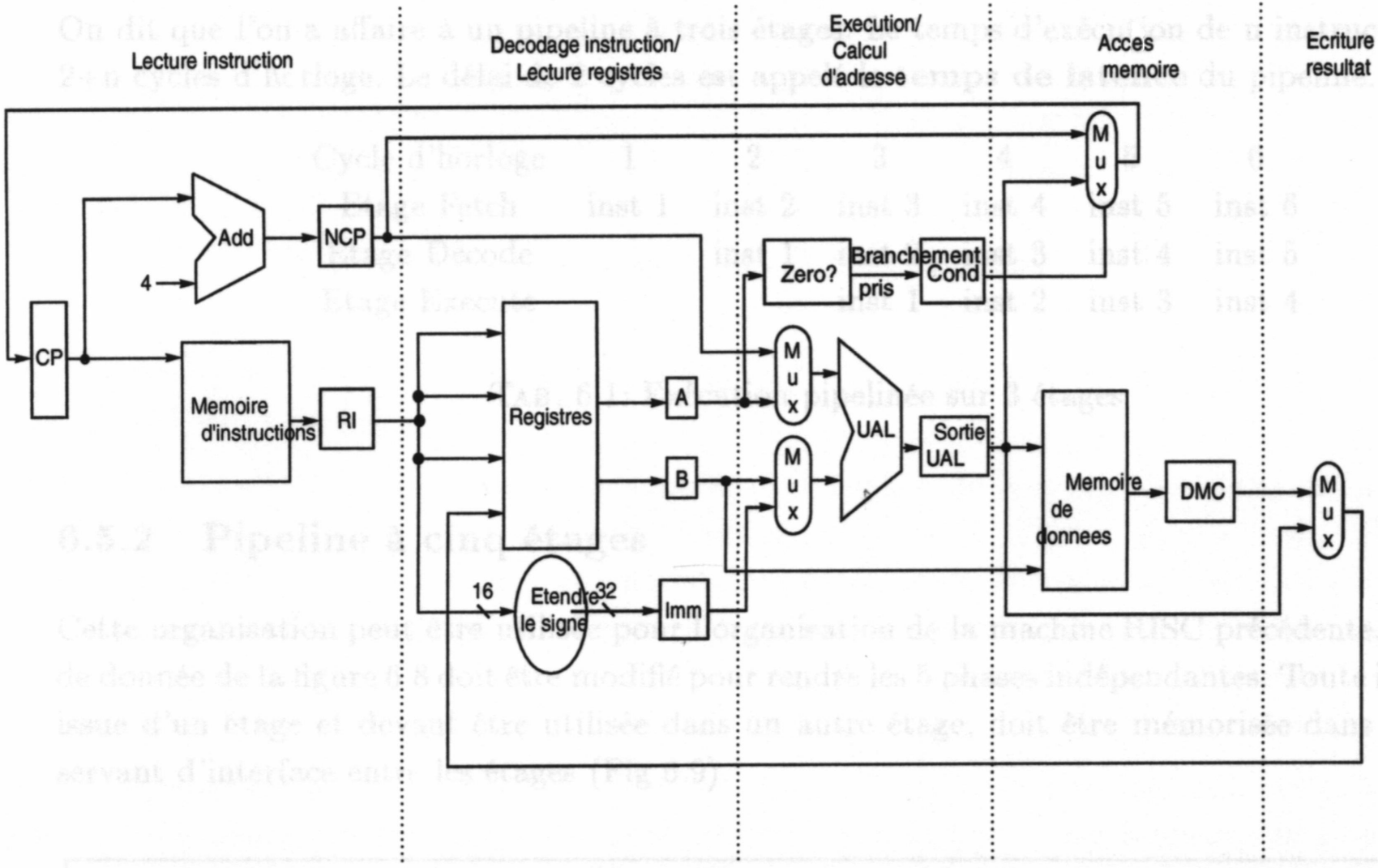
- **écriture du résultat (ER)**
  - chargement
    - registre($RI_{20..16}$) $\leftarrow$ DMC
  - opération registre-registre
    - registre($RI_{15..11}$) $\leftarrow$ SALU
  - opération registre-immédiat
    - registre($RI_{20..16}$) $\leftarrow$ SALU

performance

l'exécution d'une instruction prend 4 ou 5 cycles

les instructions de branchement sont les plus rapides

en supposant qu'elles constituent 12 % des instructions exécutés, la durée moyenne d'une instruction est de $12\% \times 4 + 88\% \times 5 = 4{,}88$ cycles.


Lecture instruction
Decodage instruction/
Lecture registres
Execution/
Calcul
d'adresse
Acces
memoire
Ecriture
resultat
Add
4
NCP
CP
Memoire
d'instructions
RI
Registres
A
B
Imm
16
Etendre
le signe
32
Zero?
Branchement
pris
Cond
M
u
x
UAL
Sortie
UAL
Memoire
de
donnees
DMC

pipeline

diminuer le temps d'exécution d'une instruction

en faisant travailler à la chaîne les différentes unités fonctionnelles

pipeline à 3 étages

3 phases *indépendantes* (fetch, decode, execute) réalisée chacune en 1 cycle d'horloge

travail en parallèle des 3 unités responsables des 3 phases

au cycle i,

– on recherche l'instruction i,
– on décode l'instruction $i-1$
– on exécute l'instruction $i-2$

pipeline à 3 étages

cycle	1	2	3	4	5	6
fetch	inst 1	inst 2	inst 3	inst 4	inst 5	inst 6
decode		inst 1	inst 2	inst 3	inst 4	inst 5
execute			inst 1	inst 2	inst 3	inst 4

temps moyen d'exécution d'une instruction divisé par 3

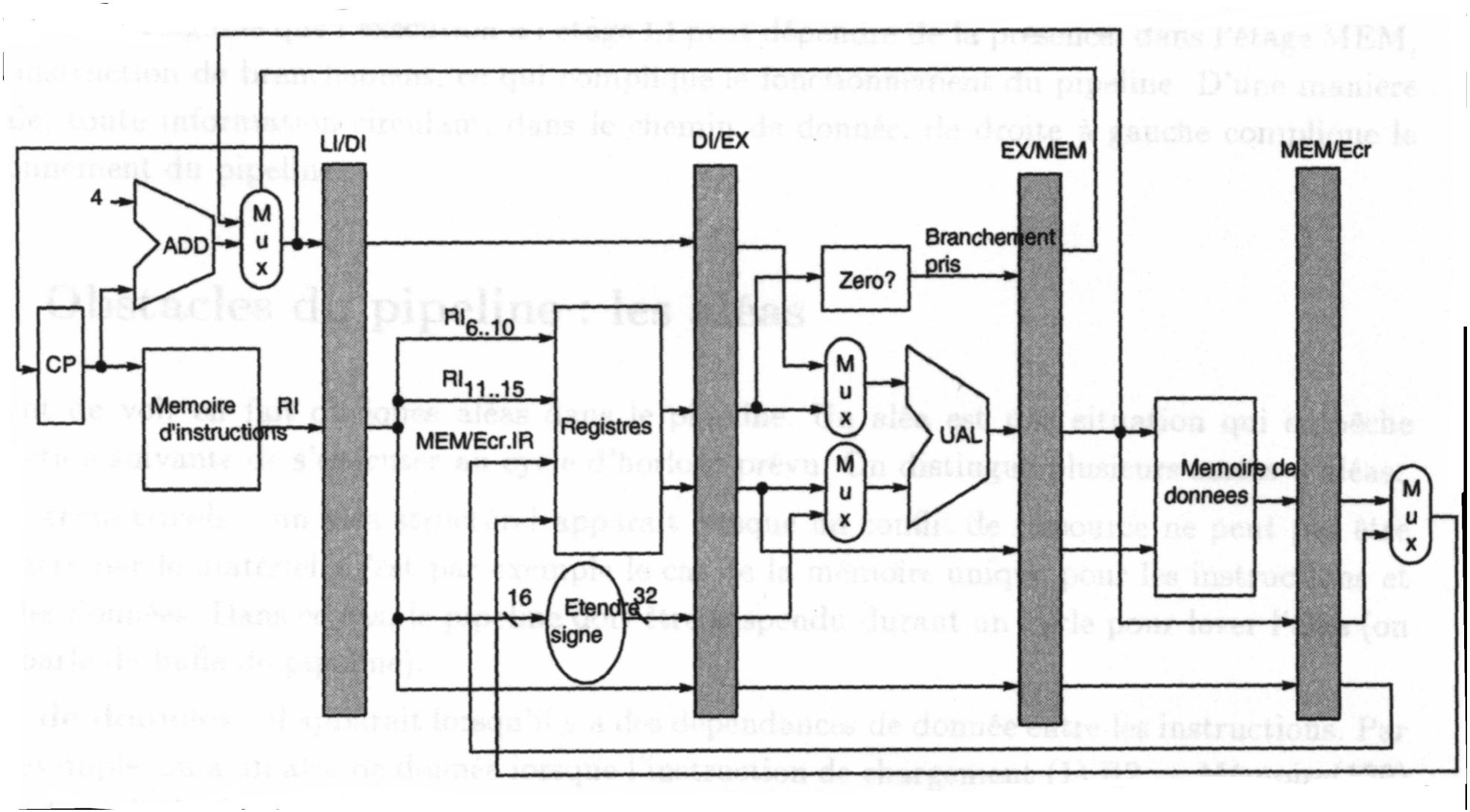
temps d'exécution de n instructions est de 2+n cycles d'horloge

pipeline à 5 étages

nécessité de

– rendre les 5 phases indépendantes
– utiliser des registres servant d'interface entre les étages

cycle	1	2	3	4	5	6
LI	inst 1	inst 2	inst 3	inst 4	inst 5	inst 6
DI		inst 1	inst 2	inst 3	inst 4	inst 5
EX			inst 1	inst 2	inst 3	inst 4
MEM				inst 1	inst 2	inst 3
ER					inst 1	inst 2


LI/DI
DI/EX
EX/MEM
MEM/Ecr
4
ADD
M u x
CP
Memoire d'instructions
RI
RI6..10
RI11..15
MEM/Ecr.IR
Registres
16
Etendre signe
32
Zero?
Branchement pris
M u x
M u x
UAL
Memoire de donnees
M u x

remarques

1. cette réalisation implique l'emploi de 2 caches, chaque cache mémoire est sollicité à chaque cycle
2. le bloc registres doit permettre deux lectures et une écriture dans le même cycle
3. le multiplexeur de sélection du registre CP a été déplacé de l'étage MEM vers l'étage LI

performance

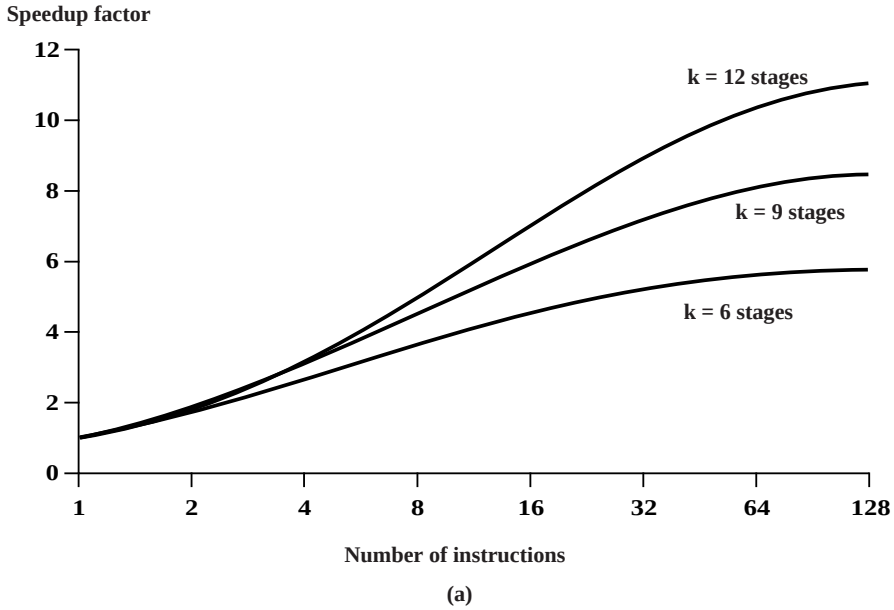
exécuter n instruction dans un pipeline à k étages

– nombre de cycle :

$$T_k = k + (n - 1)$$

– accélération :

$$\frac{T_1}{T_k} = \frac{nk}{k+(n-1)}$$



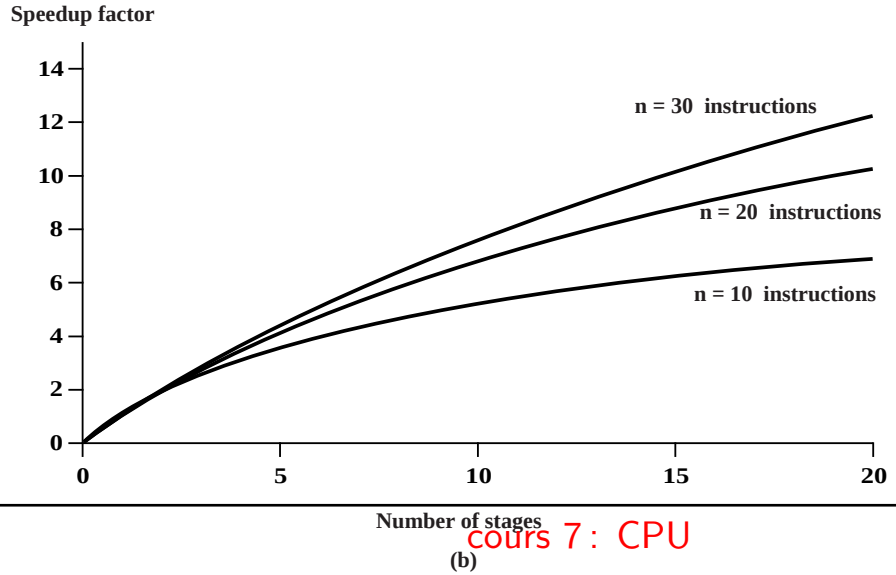


Figure 11.14 Speedup Factors with Instruction Pipelining

aléas

situation qui empêche l'instruction suivante de s'exécuter au cycle d'horloge prévu

– aléa structurel
– aléa de données
– aléa de contrôle

aléa structurel

conflit de ressources ne pouvant pas être géré par le matériel

aléa structurel

conflit de ressources ne pouvant pas être géré par le matériel

exemple : mémoire unique pour instructions et données

aléa structurel

conflit de ressources ne pouvant pas être géré par le matériel

exemple : mémoire unique pour instructions et données

solution : suspendre le pipeline durant un cycle

aléa de données

dépendance de données entre les instructions

aléa de données

dépendance de données entre les instructions

exemple

I1	r2 $\leftarrow$ mémoire(120)	(EX)
I2	r3 $\leftarrow$ r2 + r1	(DI)

aléa de données

dépendance de données entre les instructions

exemple

I1 r2 $\leftarrow$ mémoire(120) (EX)
I2 r3 $\leftarrow$ r2 + r1 (DI)

solution : intercaler une instruction vide (NOP, No OPeration) entre I1 et I2

aléa de données

le code est réécrit en

I1	r2 $\leftarrow$ mémoire(120)	(WR)
I1'	NOP	(EX)
I2	r3 $\leftarrow$ r2 + r1	(DI)

aléa de données

solution : technique d'envoi

passage direct d'un résultat à l'unité fonctionnelle qui en a besoin

pas possible dans tous les cas

aléa de contrôle

modification du registre PC

aléa de contrôle

modification du registre PC

exemple : une instruction de branchement conditionnel

jusqu'à l'exécution d'une telle instruction, il est impossible de savoir si un branchement est effectué ou non

aléa de contrôle

modification du registre PC

exemple : une instruction de branchement conditionnel

jusqu'à l'exécution d'une telle instruction, il est impossible de savoir si un branchement est effectué ou non

principal facteur de dégradation de performance dans une architecture pipeline

aléa de contrôle

cycle	1	2	3	4	5	6	7	8
instruction 1	LI	DI	EX	MEM	ER			
instruction 2		LI	DI	EX	MEM	ER		
instruction 3			LI	DI	EX	MEM	ER	
instruction 4				LI	DI	EX		
instruction 5					LI	DI		
instruction 6						LI		
instruction 7							LI	DI
instruction 8								LI

aléa de contrôle

solutions

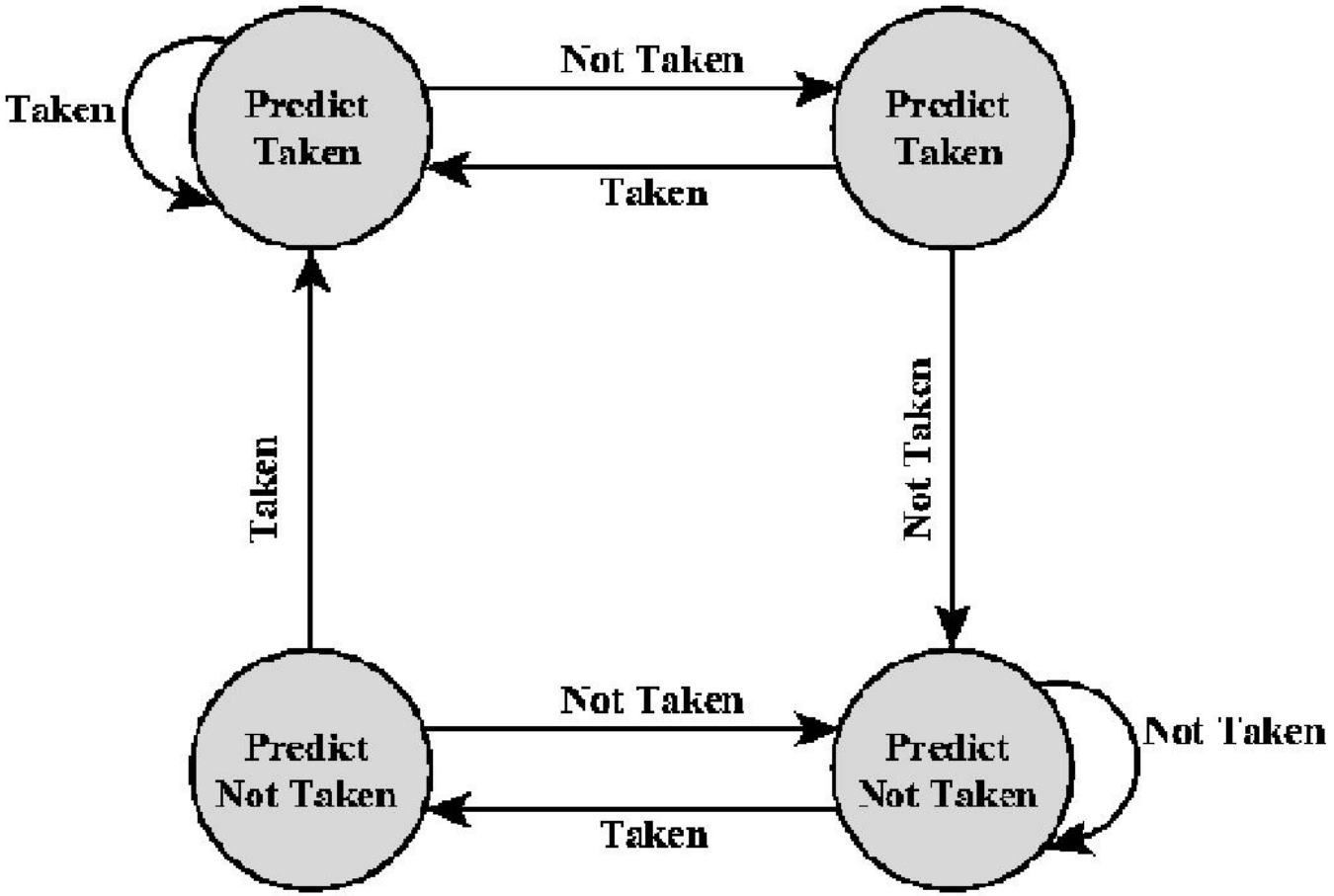
- dupliquer l'architecture de pipeline pour traiter les deux cas du branchement
- précharger l'instruction correspondant à l'adresse de branchement
- utiliser un petit cache d'instructions spécifique au fetch
- générer des instructions NOP après l'instructions de branchement

aléa de contrôle

solutions

- se baser sur une prédiction des branchements
  - supposer qu'un branchement ne sera jamais/toujours pris
  - supposer que certains opcodes favorisent le branchement
  - se baser sur un historique des branchements

prédiction


Taken
Predict Taken
Not Taken
Predict Taken
Taken
Taken
Not Taken
Predict Not Taken
Not Taken
Predict Not Taken
Not Taken
Taken

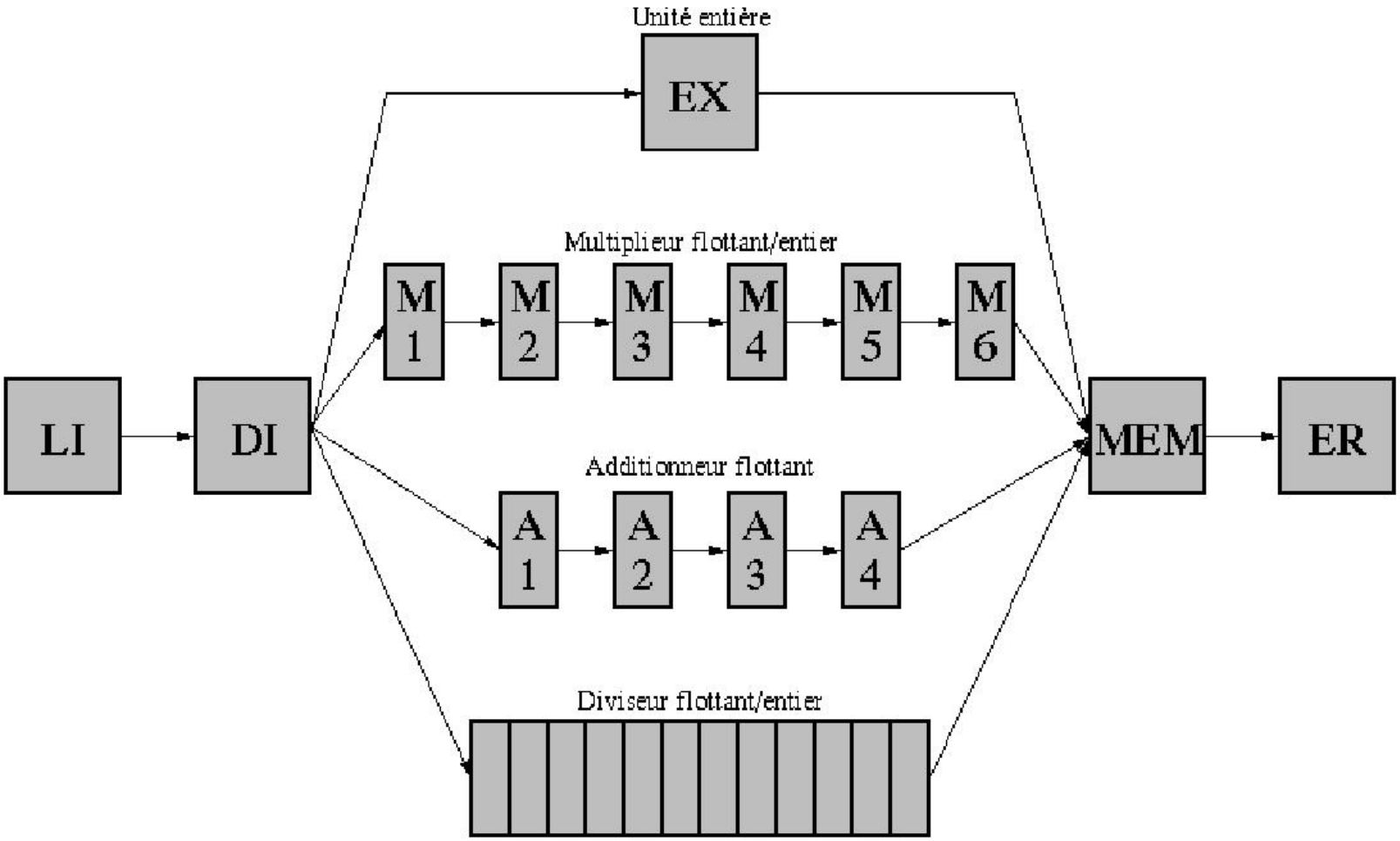
architecture superscalaire

utilisation de plusieurs unités fonctionnelles (entières, flottantes, ...)

chaque unité est pipelinée

exécution des instructions dans un ordre différent de l'ordre du programme

exemple


Unité entière
EX
Multiplieur flottant/entier
M 1
M 2
M 3
M 4
M 5
M 6
LI
DI
Additionneur flottant
A 1
A 2
A 3
A 4
MEM
ER
Diviseur flottant/entier

exemple

Par exemple, si on exécute la séquence d'instructions

– DIV x,y
– ADDF z,w
– SUB u,v

les instructions se termineront dans l'ordre SUB, ADDF et DIV.

exemple : PII

emploi d'un pipeline à 11 étages

1. fetch des instructions depuis la mémoire dans l'ordre du programme
2. traduction des instructions en une ou plusieurs instructions RISC de longueur fixe
3. exécution des instructions sur une architecture superscalaire
4. affectation des registres dans l'ordre du programme original

pipeline du PII

- Instruction Fetch Unit 1, 2 et 3
- Instruction Decode 1 et 2
- Register Allocator
- ReOrder Buffer
- Dispatcher
- Execute
- Retire Unit 1 et 2

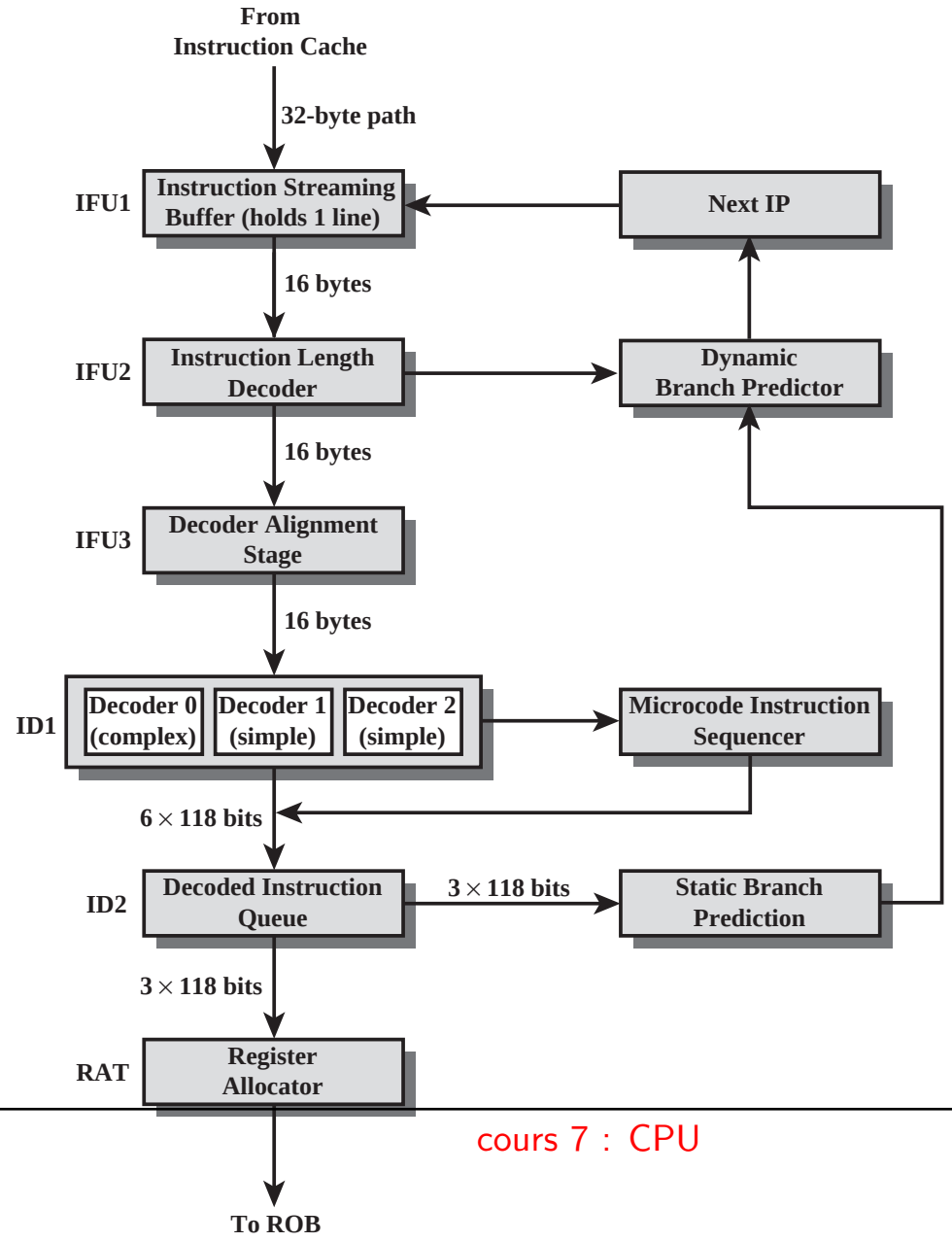


Figure 13.8 Pentium II Fetch/Decode Unit

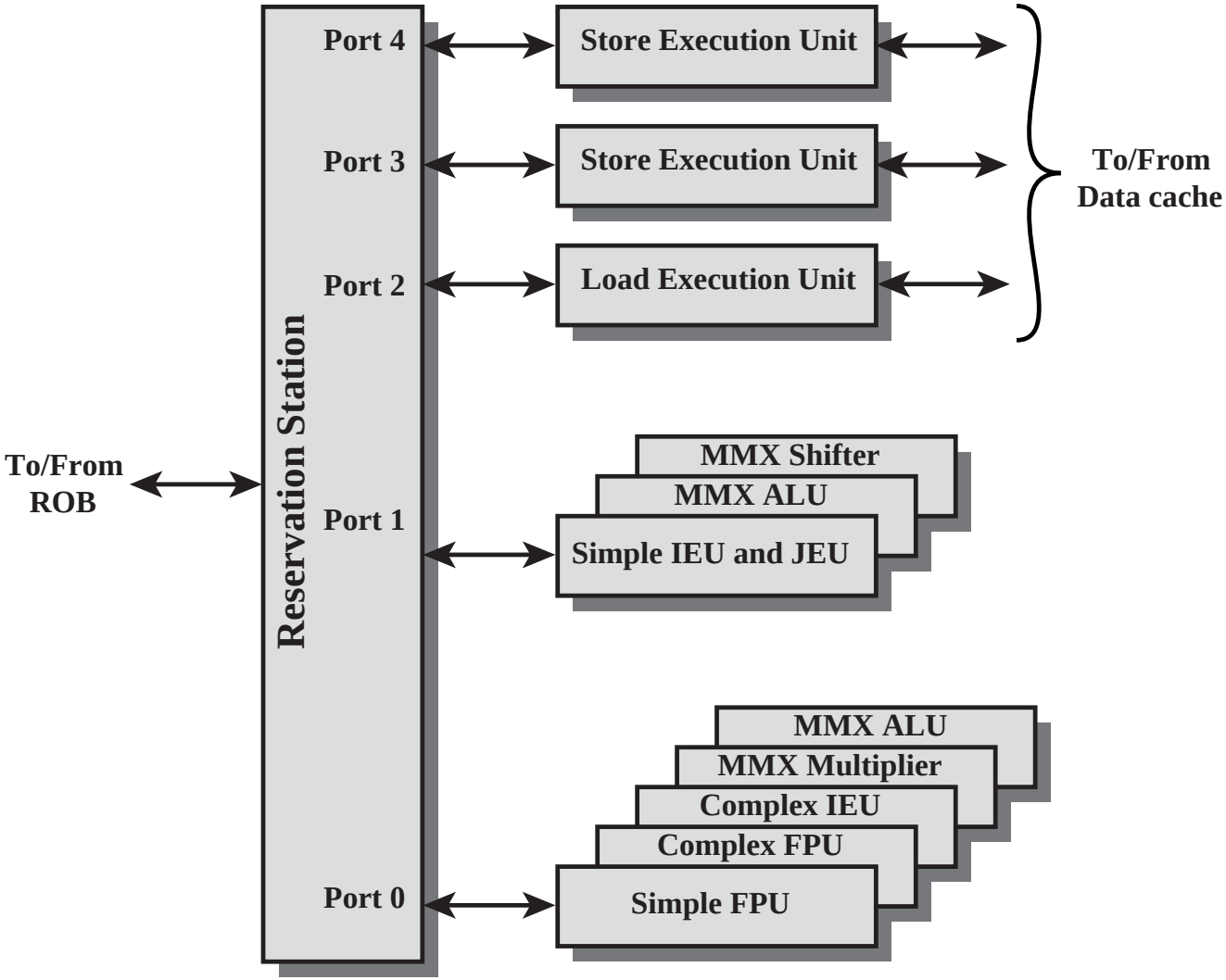
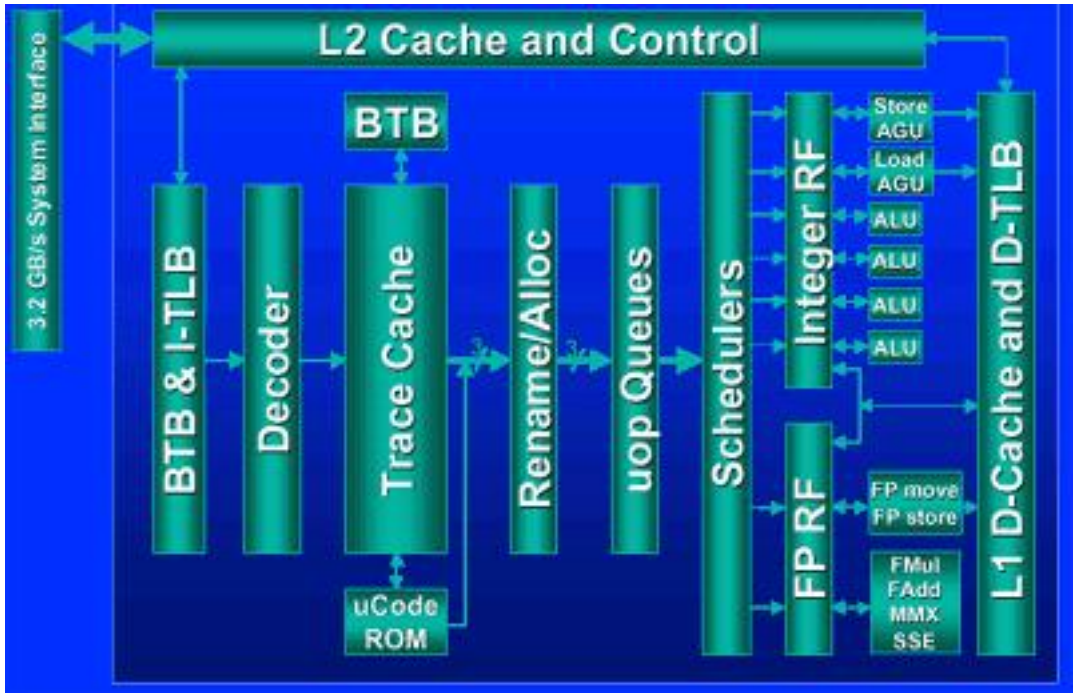


Figure 13.9 Pentium II Dispatch/Execute Unit

pipeline du PIV



architecture VLIW

Very Long Instruction Word

instruction très longue (jusqu'à 1024 bits)

composée de plusieurs champs

chaque champs commande une opération sur une unité de la machine

principe adopté dans la nouvelle architecture 64 bits développée par Intel et HP (IA 64)

exemple

une machine VLIW dispose

– d'une mémoire permettant deux opérations simultanées
– un additionneur
– un soustracteur
– un multiplieur
– un diviseur

exemple

les instructions peuvent comporter 6 champs

– 2 pour les opérations de chargement/rangement en mémoire
– 4 pour les opérations arithémtiques

exemple

soit le programme à exécuter

- A = J + K
- B = L × M
- C = N - B
- D = A / B

exemple

I1	I2	I3	I4	I5
LOAD Rj,J	LOAD Rl,L	STORE Ra,A	STORE Rb,B	STORE Rc,C
LOAD Rk,K	LOAD Rm,M	LOAD Rn,N		STORE Rd,D
	Ra = Rj + Rk			
			Rc = Rn - Rb	
		Rb = Rl × Rm		
			Rd = Ra / Rb	

ce qui fait 5 instructions pouvant s'exécuter chacune en 1 cycle d'horloge.