jeu d'instructions

lien étroit entre

– jeu d'instructions d'une machine
– sa programmation
– la conception de sa CPU

jeu d'instructions

le jeu d'instructions offre une vue logique de la machine

– registres
– types de données
– fonctionnement de l'ALU

plan

- définition
- caractéristique
  - classification
  - type d'instructions
  - type d'opérandes
  - exemple

plan

- définition
- caractéristique
  - classification
  - type d'instructions
  - type d'opérandes
  - exemple
- structure
  - adressage
  - format
  - exemple
- conclusion

rôle de la CPU

– recherche d'une instruction en mémoire
– décodage de l'instruction
– *recherche des opérandes*
– exécution de l'opération
– *stocke le résultat*

composition de la CPU

- registres
- ALU
- UC
- bus

définition

ensemble des instructions exécutables par une CPU

une instruction doit contenir

– le code de l'opération à exécuter ($opcode$)
– la référence aux opérandes sources
– la référence à l'opérande résultat
– la référence à la prochaine instruction

certaines de ces informations peuvent être implicites

représentations de l'instruction

séquence de bits découpée en champs

représentation symbolique

- l'opcode est représenté par une abbréviation (*mnémonique*)
- les registres sont représentés par leur nom
- les adresses mémoires par leur valeur ou un nom de variable
- des symboles particuliers pour les contenus, les modes d'adressage, ...

exemple

adresse mémoire	contenu			
0100	0010	0010	0000	1100
0101	0001	0010	0000	1101
0110	0001	0010	0000	1110
0111	0011	0010	0000	1111
1100	0000	0000	0000	0010
1101	0000	0000	0000	0011
1110	0000	0000	0000	0100
1111	0000	0000	0000	0000

exemple

adresse	contenu	
0100	LOAD	(1100)
0101	ADD	(1101)
0110	ADD	(1110)
0111	STORE	(1111)

1100	0002
1101	0003
1110	0004
1111	0000

langages

compilation d'un langage haut niveau (HLL) vers un langage de bas niveau (LLL)

une instruction HLL correspond à plusieurs instructions LLL

le jeu d'instruction doit être suffisament expressif pour coder toute instruction d'un langage de haut niveau

exemple

$X = X + Y$ se traduit en

```
LOAD    X,   R1
ADD     R1,  Y
STORE   R1,  X
```

conception d'un jeu d'instruction

compromis entre

- nombre d'opérations
- complexité des opérations
- nombre de types de données
- nombre de registres
- utilisation des registres
- nombre de modes d'adressage
- nombre de champs
- taille des champs

classification

selon le format d'instruction

selon la relation entre mémoire et registres

format d'instruction

nombre de champs réservés aux adresses

en théorie

– premier opérande source
– deuxième opérande source
– opérande résultat
– prochaine instruction

en pratique

- format 3 adresses : peu courant (instructions longues)
- format 2 adresses : une adresse fait office de source et de destination
- format 1 adresse : l'accumulateur garde une opérande et le résultat
- format 0 : une pile stocke opérandes et résultats

nombre d'adresses

nombre d'adresses	représentation	interprétation
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	ACC ← ACC OP A
0	OP	T ← T OP T - 1

ACC : l'accumulateur

T : sommet de pile

choix du nombre d'adresses

moins il y a d'adresses

plus les instruction sont courtes

moins la CPU est complexe

plus les instructions sont nombreuses

plus les programmes sont lents à exécuter

choix du nombre d'adresses

moins il y a d'adresses

plus les instruction sont courtes

moins la CPU est complexe

plus les instructions sont nombreuses

plus les programmes sont lents à exécuter

actuellement : mélange des formats 2 adresses et 3 adresses

relation mémoire-registres

disposer de registres permet de minimiser les accès à la mémoire et d'accélérer l'exécution

- registre-registre (chargement-rangement)
  - opérations LOAD et STORE
  - instructions simples
  - exécution rapide (nombre de cycles fixe)
  - nombre d'instructions générées important

relation mémoire-registres

- registre-mémoire
  - code généré compact
  - instructions plus difficiles à décoder
  - nombre de cycles pour l'exécution est variable
- mémoire-mémoire
  - la mémoire peut devenir un goulôt d'étranglement

actuellement : chargement-rangement

catégories d'instructions

- transfert de données
- opérations arithmétiques
  - entiers signés
  - flottants
- logique
- conversion
- E/S
- contrôle système
- transfert de contrôle (test et branchement)
  - test du registre condition

exemple

i = 0;

while(i < 10) i ++;

```
        ...
        STORE i, 0
        LOAD R1, 10
        LOAD R2, i
 N1     ADD R2, R2, 1
        SUB R3, R1, R2
        JNZ N1
        ...
```

catégories d'opérandes

- adresses
- nombres
  - entiers (fixed-point)
  - flottants (floatting point)
  - DCB
- caractères (ASCII)
- données logiques

voire des types de données plus évoluées (liste, chaîne de caractères)

exemple du pentium II

données de longueur 8, 16, 32 ou 64 bits

- contenu arbitraire sur 8, 16, 32 ou 64 bits
- entier en complément à 2 sur 8, 16 ou 32 bits
- entier non signé sur 8, 16 ou 32 bits
- BDC sur 8 bits
- packed BCD : 2 chiffres BDC (de 0 à 99) sur 8 bits
- adresse sur 32 bits
- séquence de bits indépendants
- séquence d'octets, mots ou doubles mots

pII : données

- flottants : types utilisés par l'unité en virgule flottante (FPU)
  - entier en complément à 2 sur 16, 32 ou 64 bits
  - packed BCD : 1 bit de signe, 18 octet BDC
  - standard IEEE 754 :
    - simple précision
    - double précision
    - précision étendue (64 bits de mantisse, 15 bits d'exposant)

exemple : Power PC

données de longueur 8, 16, 32, 64 bits

– entier non signé sur 8, 16 ou 32 bits
– entier signés sur 16 ou 32 bits
– adresse sur 32 ou 64 bits
– séquence d'octets (bornée à 128 octets)
– IEEE 754 simple et double précision

PII : opérations

jeu d'opérations complexe

nombreuses opérations spécialisées

optimiser la traduction de code de haut niveau

en plus des opérations conventionnelles

– support des opérations de branchement de langage de haut niveau
– gestion de la mémoire et du cache interne
– nombreuses manières de tester le registre de condition

PII : instructions MMX

– introduites en 1996
– 57 nouvelles instructions
– dédiées aux opérations multimédia

principe *Single Instruction Multiple Data*

données vidéo et audio composées de large tableaux de petits types de données (souvent 8 ou 16 bits)

PII : MMX

trois nouveaux types de données de 64 bits découpés en champs

– packed bytes : 8 octets
– packed word : 4 mots de 16 bits
– packed doubleword : 2 double mots de 32 bits

les nouvelles instructions opèrent en parallèle sur chacun des champs

quelques modes d'adressage classiques

- l'adressage immédiat
  - pas de référence mémoire
  - taille de l'opérande limitée
- l'adressage direct
  - simple
  - taille de l'espace adressable limitée
- l'adressage registre
  - pas de référence mémoire
  - taille de l'espace adressable limitée

adressage

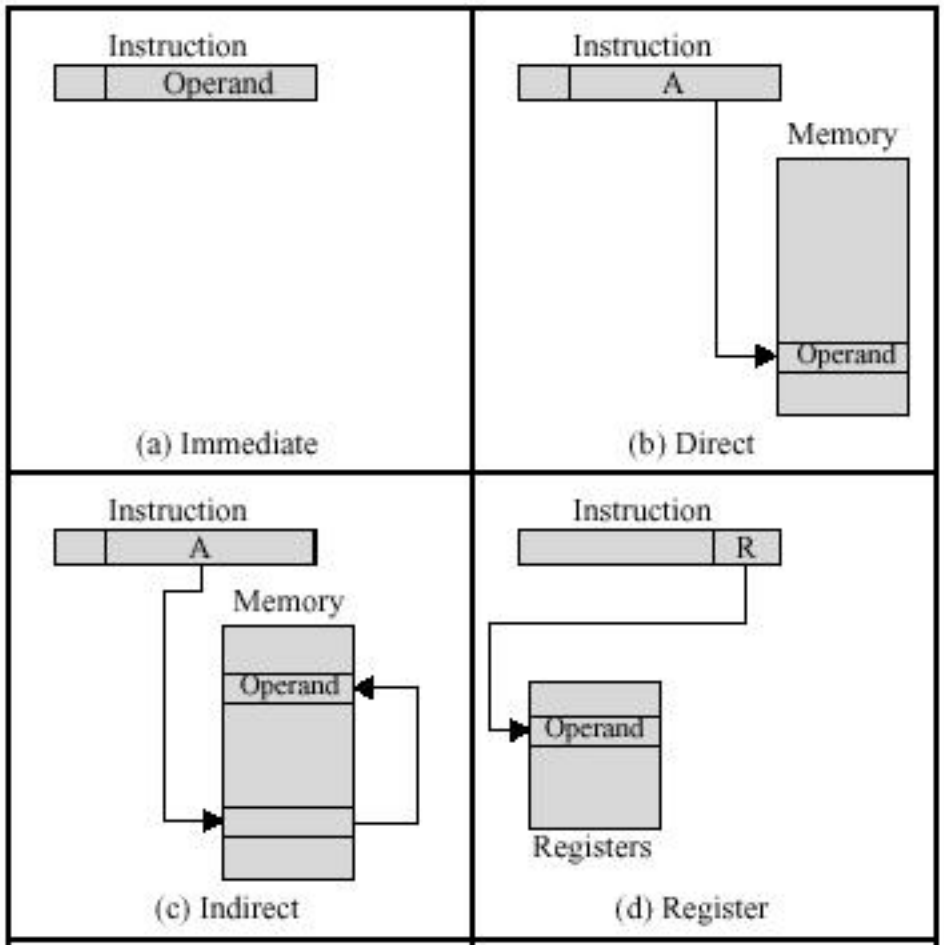
comment adresser un espace mémoire important en conservant une taille d'instruction raisonnable ?

adressage

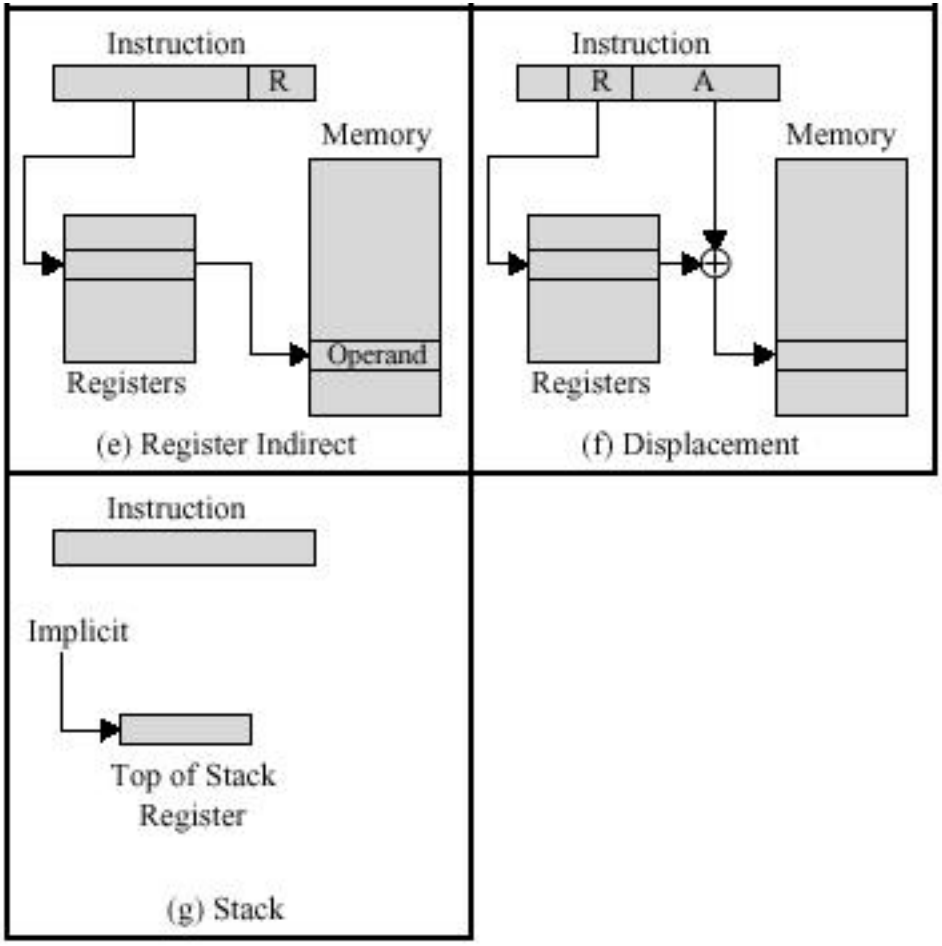
comment adresser un espace mémoire important en conservant une taille d'instruction raisonnable ?

- l'adressage indirect par mémoire
  - plusieurs références à la mémoire
- l'adressage indirect par registre
- adressage avec déplacement
  - flexible
  - complexe

modes d'adressage



modes d'adressage



adressage

registre	ADD R4, R3
immédiat	ADD R4, 3
direct	ADD R4, (0011)
indirect par registre	ADD R4, (R3)
indirect par mémoire	ADD R4, @(0011)
avec déplacement	ADD R4, (R3)100

adressage

antagonisme entre

– espace adressable et flexibilité
– nombre de références mémoire et complexité du calcul d'adresse

disposer de plusieurs modes d'adressage

disposer de registres spécifiques

les modes immédiat, indirect par registre et avec déplacement sont les plus utilisés

adressage

2 manières de différencier les modes d'adressage

- utilisation d'un ou plusieurs bits (spécificateur d'adresse)
  - utile pour disposer d'un grand nombre de modes d'adressage
  - peut donner lieu à une taille d'instruction variable
- utilisation d'opcodes différents
  - permet de conserver une taille d'instruction fixe
  - facilite la réalisation matérielle

taille d'instruction

plus le jeu d'instructions est complexe

plus la quantité d'instructions dans un programme est faible

plus la longueur d'instruction est importante

plus la consommation en espace est importante

taille d'instruction

La longueur d'une instruction doit prendre en considération

– la taille et l'organisation de la mémoire
– la structure d'interconnexion
– la complexité et la vitesse de la CPU

taille d'instruction

la taille de l'instruction peut être

- fixe
- variable
  - autorise une large gamme d'opcodes (de tailles différentes)
  - une plus grande flexibilité d'adressage
  - accroit la complexité de la CPU

allocation des bits

le découpage en champs dépend

– du nombre d'opérandes
– du nombre d'opérations
– du nombre de modes d'adressage
– de l'utilisation des registres
– de l'espace adressable
– de la façon d'adresser la mémoire

taille de l'opcode

si on souhaite à la fois

– une taille d'instruction raisonnable
– une capacité d'adressage raisonnable
– un nombre d'opcodes important

on peut utiliser une taille d'opcode variable

code opération étendu

40 opcodes dont seulement 15 nécessitent un paramètre de 12 bit

code opération 4 bits	paramètres 12 bits

code opération 4 bits	code opération étendu 5 bits	non utilisé 7 bits

16 bits au lieu de 18 bits

registres

utilisés pour recevoir données et adresses

espace d'adressage nécessitant quelques bits d'adresse

souvent séparés en groupes (données, adresses) identifiés au niveau de l'opcode

exemple : l'ALPHA de DEC

– 32 registres de 64 bits pour la manipulation des entiers
– 32 registres de 64 bits pour la manipulation des flottants

instructions de taille fixe (32 bits)

formats d'instruction

4 formats d'instruction

– instructions spécifiques au système d'exploitation
– branchement
– transfert de données
– opérations de calcul entier ou flottant

ALPHA

opcode 6 bits	nombre 26 bits

opcode 6 bits	Ra 5 bits	déplacement 21 bits

opcode 6 bits	Ra 5 bits	Rb 5 bits	déplacement 16 bits

opcode 6 bits	Ra 5 bits	Rb 5 bits	fonction 11 bits	Rb 5 bits

exemple : le SPARC de SUN

grand nombre de registres (> 520)

32 registres visibles simultanéments

4 groupes de registres spécifiques

instructions de taille fixe (32 bits)

exemple : le SPARC de SUN

3 formats d'instructions (format codé sur 2 bits)

- appel de sous-programme
- branchement ou chargement d'une donnée dans les poids fort d'un registres
- autres opérations au format 3 adresses

SPARC

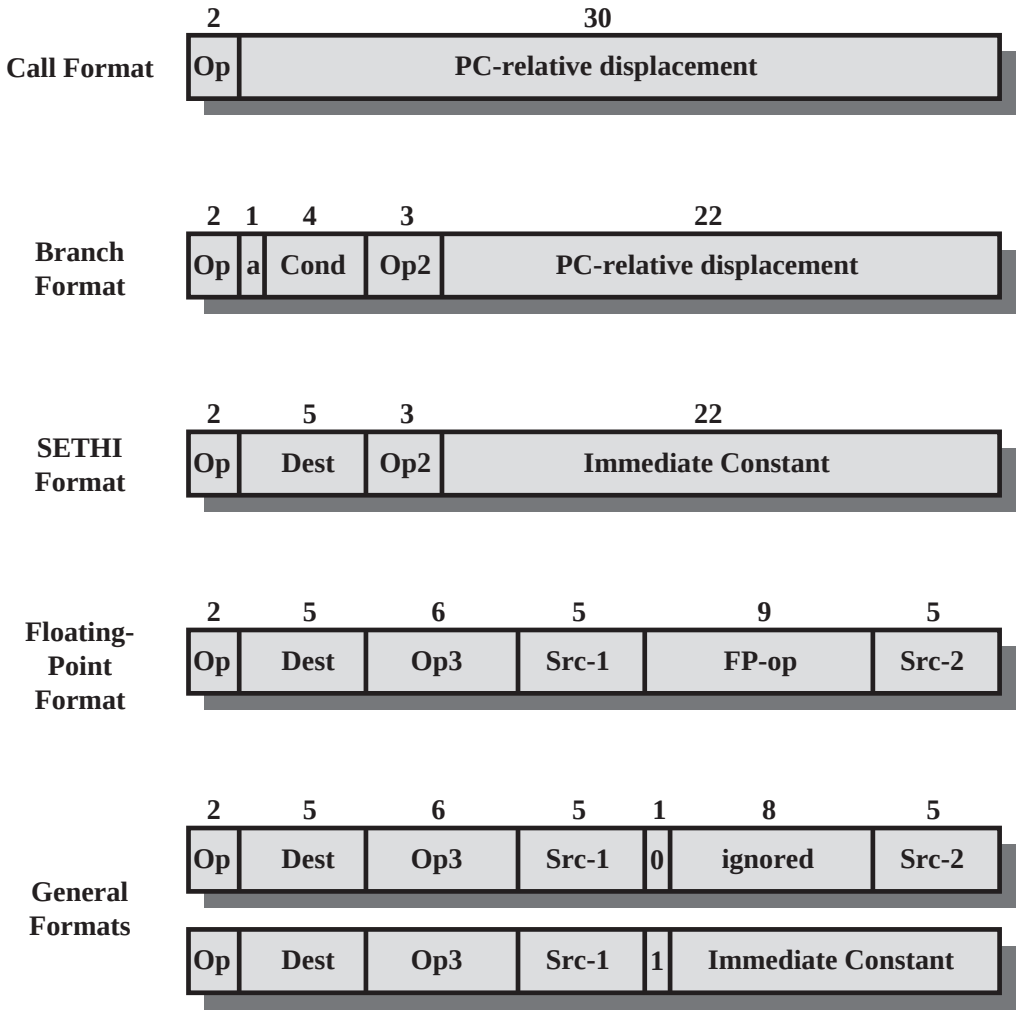


Figure 12.13 SPARC Instruction Formats

exemple : le PowerPC

9 modes d'adressages classés selon le format d'instruction

format	mode
chargement-rangement	indirect
	indirect indexé
branchement	absolu
	relatif au PC
	indirect
calcul entier	registre
	immédiat
calcul flottant	registre

exemple : le PII d'INTEL

– 8 registres généraux 32 bits
– 8 registres généraux 16 bits
– 8 registres généraux 8 bits

2 registres 32 bits sont utilisés pour les opérandes 64 bits

PII

un adresse = un ségment + un offset

- 6 registres de ségment
- 6 registres de description de ségment
- 1 registre de base
- 1 registre d'index

PII

9 modes d'adressage

– immédiat

– registre

– déplacement

contenu du registre ségment

\+ déplacement inclu dans l'instruction

PII

– indirect par registre

  contenu du registre ségment

  \+ contenu du registre de base

– base et déplacement

  idem précédent

  \+ déplacement contenu dans l'instruction

PII

– base indexé avec déplacement

idem précédent

\+ contenu du registre index

– base indexé avec déplacement

idem précédent

\+ contenu du registre index $\times$ un facteur

PII

– indexé avec déplacement

  idem précédent - base

– relatif

  contenu du compteur ordinal

  \+ déplacement contenu dans l'instruction

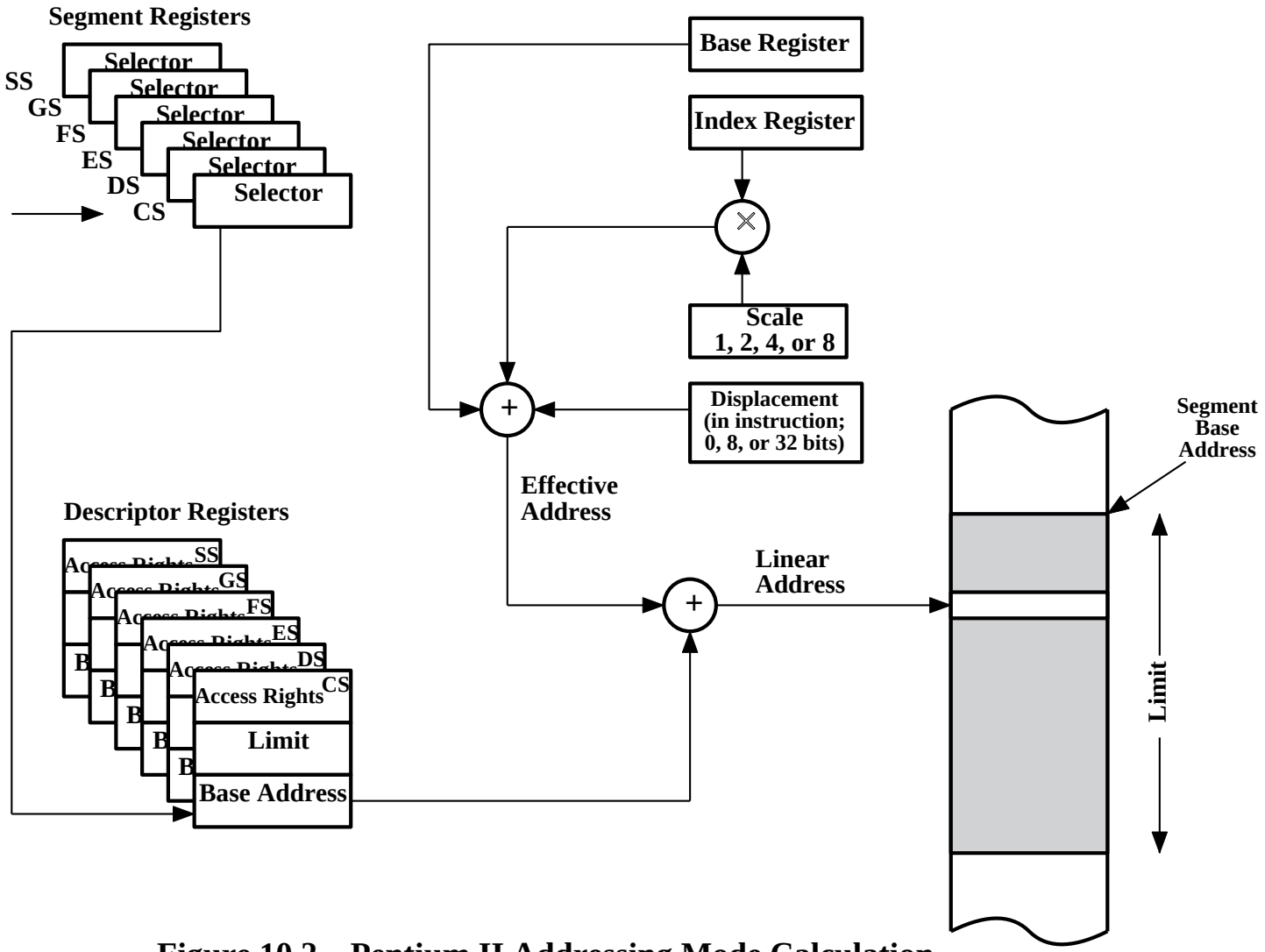


Figure 10.2 Pentium II Addressing Mode Calculation

PII

grande variété de formats

– efficacité d'exécution de langages de haut niveau
– respect de la compatibilité ascendante de la famille 8086

mode d'adressage donné par l'opcode

PII

l'instruction comporte

- un préfixe de 0, 1, 2, 3 ou 4 octet,
- un opcode de 1 ou 2 octets,
- un spécificateur d'adresse de 0, 1 ou 2 octets,
- un déplacement de 0, 1, 2 ou 4 octets,
- un champ pour l'adressage immédiat de 0, 1, 2 ou 4 octets

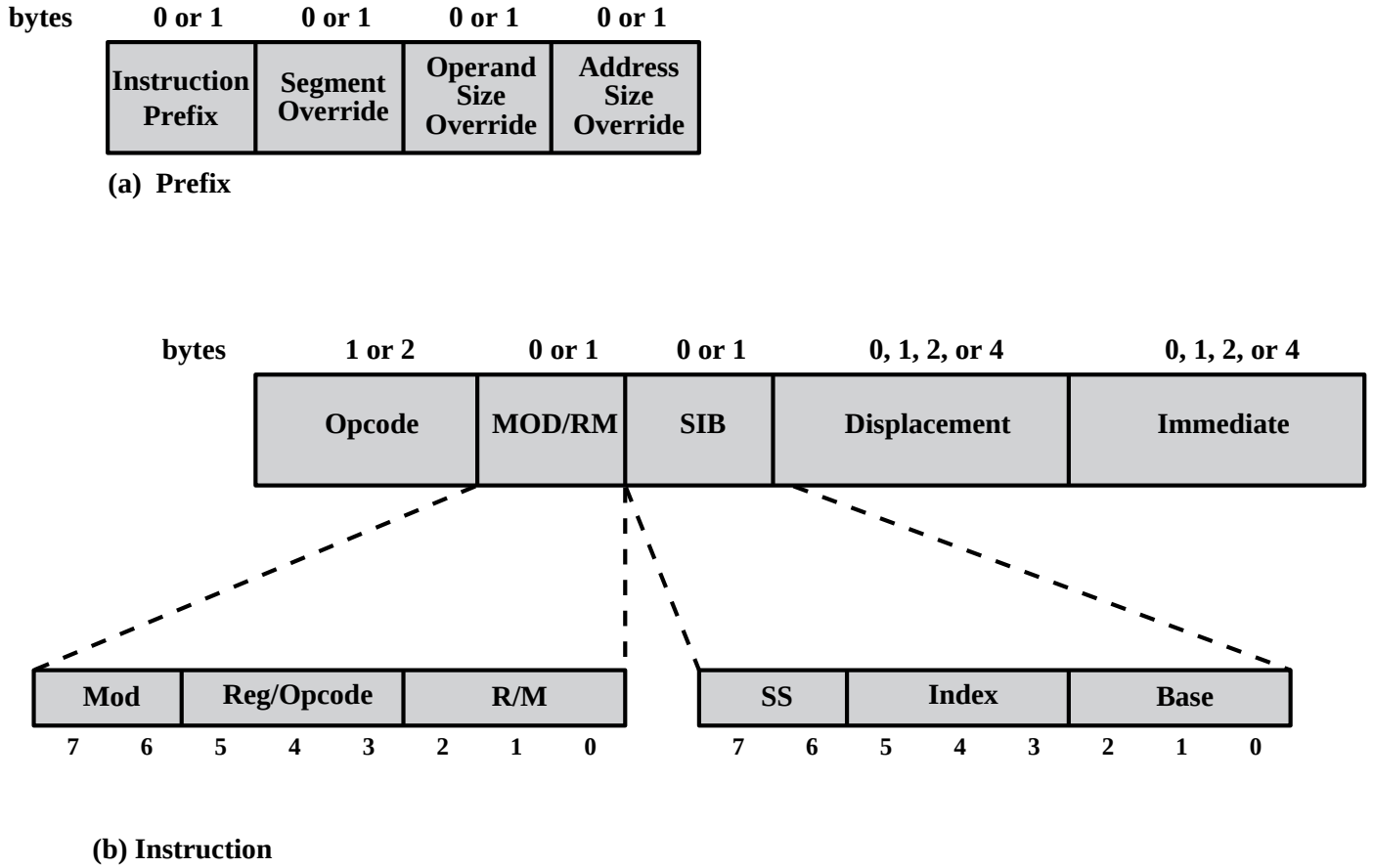


Figure 10.8 Pentium II Instruction Format

conclusion

on oppose traditionnellement

– CISC (Complex Instruction Set Computer) complexe en nombre d'instructions, modes d'adressage, registres, etc...
– RISC (Reduced Instruction Set Computer)

difficile de comparer des machines CISC à des machines RISC

les deux technologies ont convergé

exercice corrigé

but : comparer 4 architectures

– accumulateur
– mémoire-mémoire
– pile
– chargement-rangement

exercice corrigé

hypothèses :

– opcode sur 1 octet
– adresse mémoire sur 2 octets
– opérande sur 4 octets
– taille d'instruction multiple d'1 octet

accumulateur

opcode	adresse
8 bits	16 bits

```
LOAD    B
ADD     C
STORE   A
ADD     C
STORE   B
SUBINV  A
STORE   D
```

7×24 bits $=$ 168 bits

mémoire-mémoire

opcode	source 1	source 2	destination
8 bits	16 bits	16 bits	16 bits

```
ADD  B, C, A
ADD  A, C, B
SUB  A, B, D
```

3×56 bits $=$ 168 bits

pile

opcode 8 bits

opcode 8 bits	adresse 16 bits

pile

```
PUSH  B     ADD
PUSH  C     POP   B
ADD         PUSH  A
POP   A     PUSH  B
PUSH  A     SUB
PUSH  C     POP   D
```

9×24 bits $+ 3 \times 8$ bits $= 240$ bits

chargement-rangement

opcode 8 bits	Ri 4 bits	xx 4 bits	adresse 16 bits

opcode 8 bits	R1 4 bits	R2 4 bits	Rd 4 bits	xx 4 bits

chargement-rangement

LOAD	Rb, B	ADD	Ra, Rc, Rb
LOAD	Rc, C	STORE	B, Rb
ADD	Rb, Rc, Ra	SUB	Ra, Rb, Rd
STORE	A, Ra	STORE	D, Rd

5×32 bits $+ 3 \times 24$ bits $= 232$ bits

accès mémoire

– accumulateur

7 transferts de données ($7 \times 32 = 224$)

$+$ taille du code (168)

$=$ 392 bits sur 7 instructions (56 bits/instruction)

– mémoire-mémoire

9 transferts de données ($9 \times 32 = 288$)

$+$ taille du code (168)

$=$ 456 bits sur 3 instructions (152 bits/instructions)

accès mémoire

– pile

9 transferts de données ($9 \times 32 = 288$)

$+$ taille du code (240)

$=$ 528 bits sur 12 instructions (44 bits/instructions)

– chargement-rangement

5 transferts de données ($5 \times 32 = 160$)

$+$ taille du code (232)

$=$ 392 bits sur 8 instructions (49 bits/instructions)