Architecture des ordinateurs

Comment sont exécutés les programmes?

=> du langage de haut niveau au langage machine

Composants et couches d'une architecture

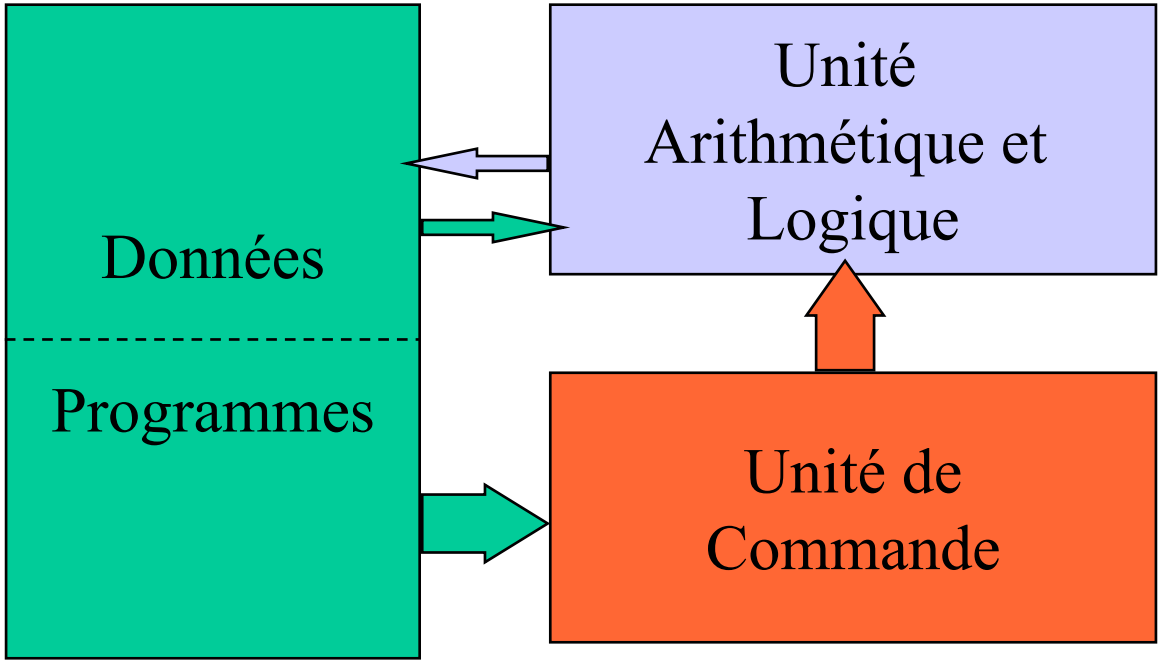
=> micro-architecture, bus, entrées-sorties, mémoires, horloge, processeurs

Facteurs d'amélioration des performances

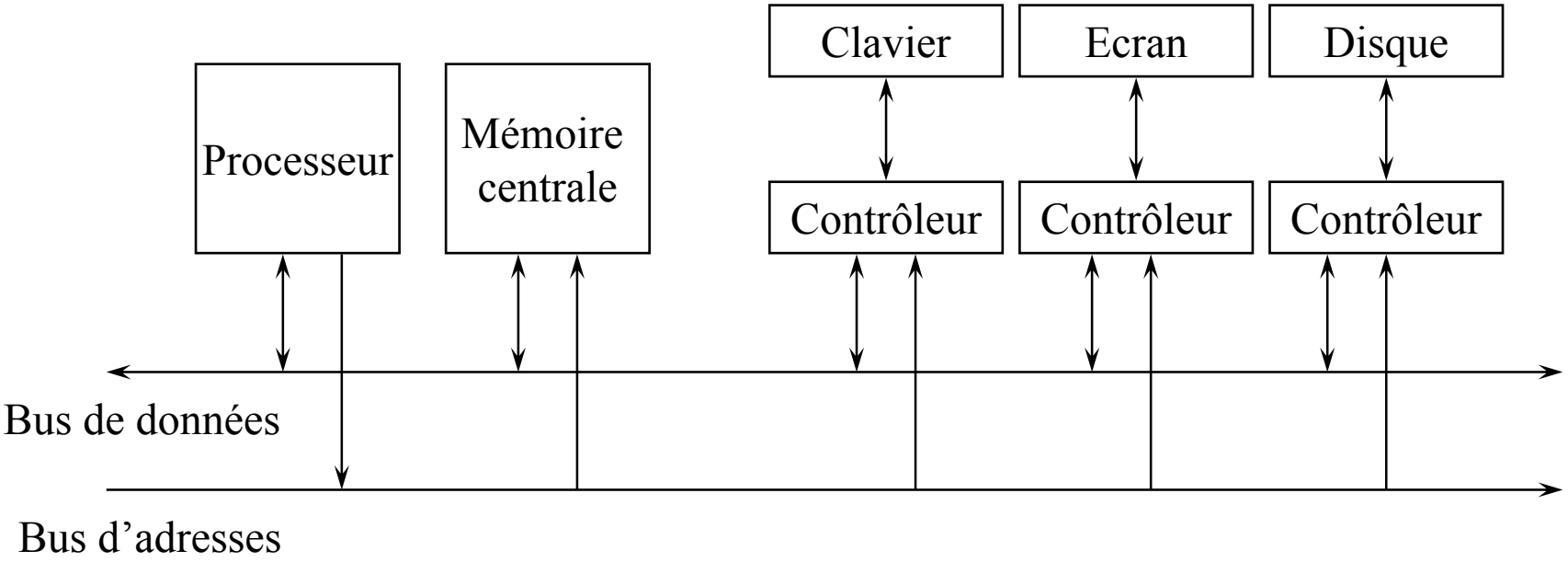
=> pipeline, cache, parallélisme

Bibliographie:
Architecture de l'ordinateur *Andrew Tanenbaum* (Dunod)
Architectures logicielles et matérielles *Paul Amblard et al*(Dunod)

Le modèle de Von Neuman



Architecture simplifiée

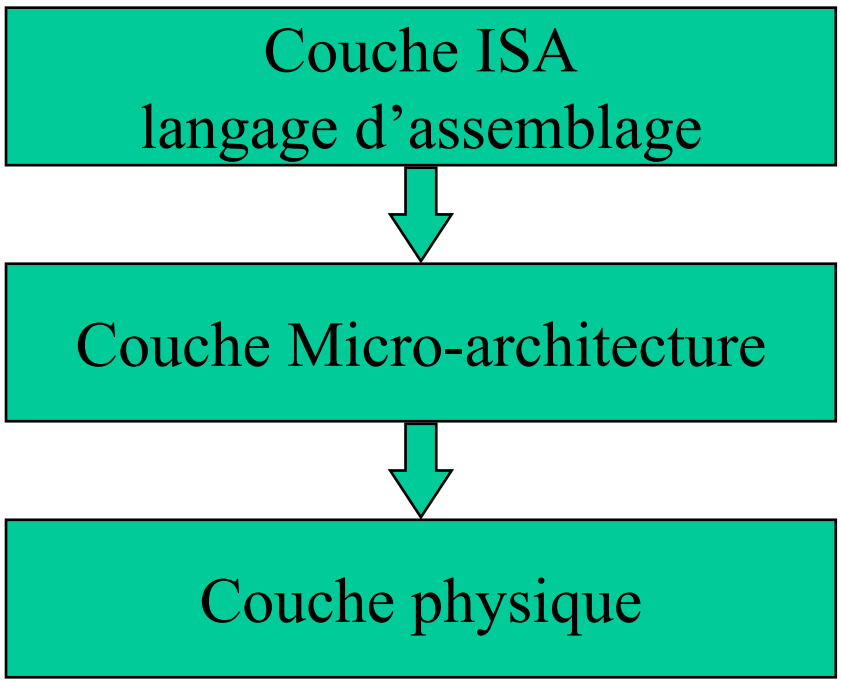


Composants de base

- Mémoire organisée en mots (emplacements numérotés)
- Processeur : unités de calcul, de commande et registres
- Bus: transfert d'information, largeur

Couches d'une architecture

compilateur/système d'exploitation



mémoire, certains registres
instructions, entrée-sorties

chemin des données, mémoires
micro-programme, composants

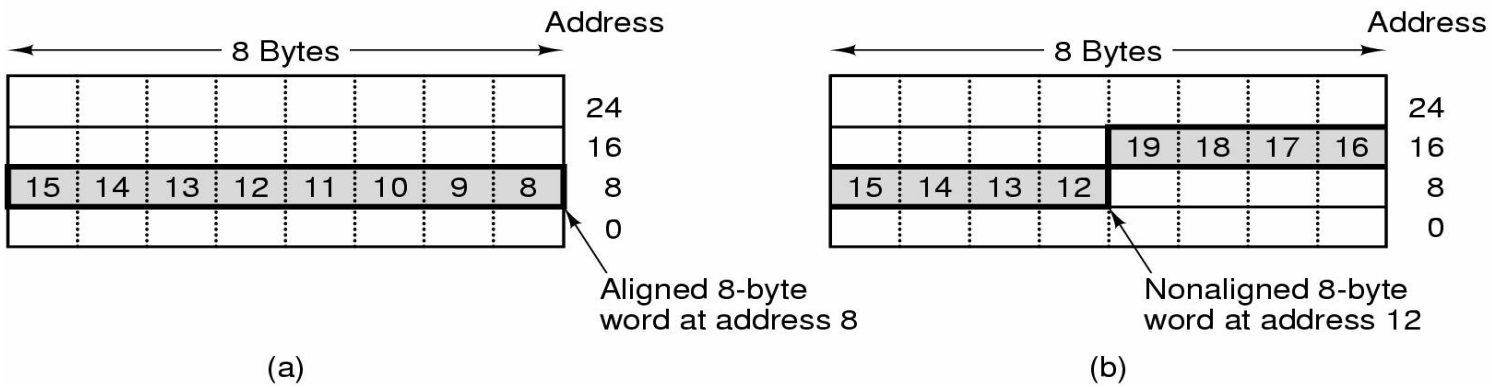
Circuits logiques

Langage d'assemblage et langage machine

- Langage machine:
  - ensemble des instructions comprises par le processeur
  - convention de codage
- Langage d'assemblage
  - description textuelle du langage machine
  - Assembleur pour traduire en langage machine

Organisation de la mémoire

- En mots (parfois plusieurs découpages co-existent).
  - Souvent mots de 8 bits (octets), 16 ou 32 bits
  - Souvent alignés



Registres

- Mémoires situées dans le processeur
  - accès rapide
  - en général un mot
  - généralistes ou spécialisés
- Exemple de registres spécialisés:
  - compteur ordinal (adresse de l'instruction exécutée)
  - registre d'état de l'UAL (PSW) NZVCAP

Codage des données

- Données numériques:
  - Entiers, signés ou non, plusieurs longueurs
  - Flottants, 32,64,128 bits
  - BCD (en Cobol)
- Données non numériques
  - Caractères (ASCII = 1 octet, Unicode= 2 octets)
  - booléen: octet
  - adresse: entier non signé

Organisation des données numériques en mémoire

Gros boutiste (big Endian): octets rangés du poids fort au poids faible

Petit boutiste (little Endian): octets rangés du poids faible au poids fort

Autres

Echange d'information difficile

PC

Modes d'adressage

- Décrit la manière dont l'accès à la donnée est codé dans l'instruction
- adressage immédiat: la donnée est codée.
- adressage direct:
  - l'adresse a du mot mémoire contenant la donnée est codée. Mem[a]
  - direct par registre: le numéro n du registre est codé. Reg n

- adressage indirect:
  - indirect par registre: numéro du registre n qui contient l'adresse du mot mémoire contenant la donnée Mem[contenu de Reg n]
  - indirect par registre avec déplacement: numéro du registre, valeur constante relative d, Mem[contenu de Reg n + d]
- adressage indexé:
  - indirect par registre avec index: deux numéros de registre n, m Mem[Reg n+ Reg m]
  - indexé basé: deux numéros de registre et un déplacement

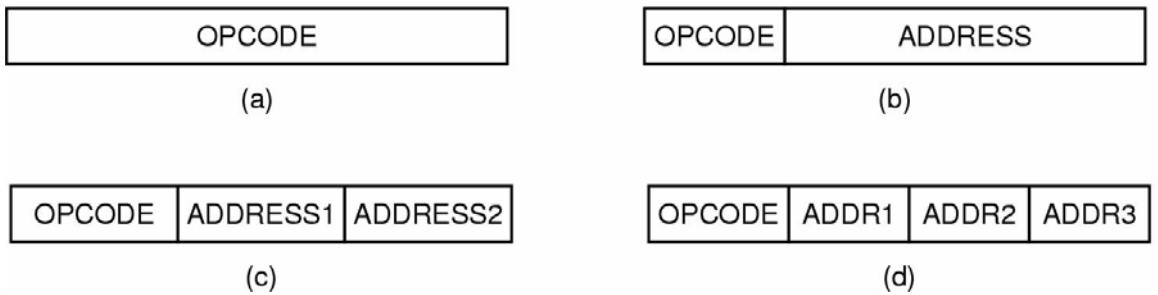
Dans certaines machines, il existe aussi l'adressage avec mise à jour:

ex adressage indirect par registre avec déplacement et mise à jour

Mem[contenu de Reg_n + dep] et Reg_n<-Reg_n+dep

Codage des instructions

- Coder quoi?
  - L'opération
  - Le mode d'adressage et les opérandes

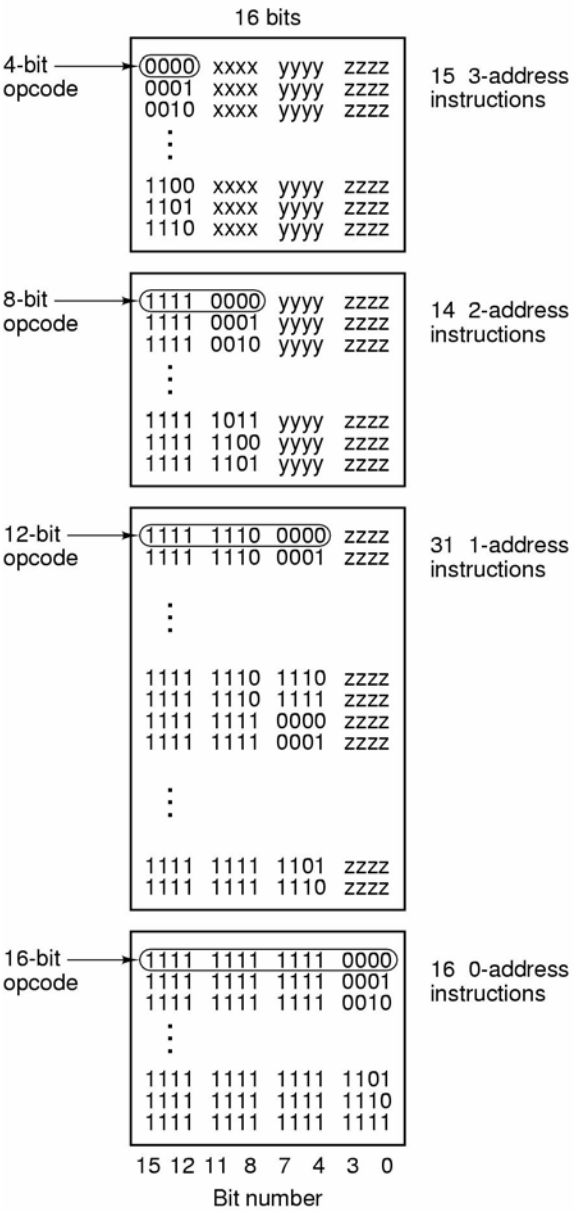


Taille fixe ou variable?

Efficacité d'un codage

- Compacité (peu de bits)
- organisation de la mémoire/taille des champs opérande
- possibilités d'extension
- Rapidité de décodage.

Code expansif



- Champs de taille variable selon l'instruction, mais code de taille fixe.
- Exemple
  - 16 bits pour instruction. adresses codées sur 4 bits, 16 registres. 15 instructions à trois adresses, 14 à 2 adresses, 31 1 adresse et 16 sans

Principales instructions

- Transfert de données: source vers destination (selon modes d'adressage)
- Opérations arithmétiques et logiques
- Branchements (spécifie la prochaine instruction si pas la suivante)
  - non conditionnels
  - conditionnels (selon bits registre PSW)

Structures de contrôle des langages de haut niveau

- If then else

```
        CMP R1,0
        BNE L1
        MOV R2,R3
        MOV R4,R5
        BR L2
L1:     MOV R6,R7
        MOV R8,R9
L2:     MOV R7,R8
```

- Boucle

```
L1:     CMP R1,0
        BE  L2
        ADD R2,R3, R2
        SUB R1,1
        BR L1
L2:     MOV R7,R8
```

Appels de procédures: la pile d'exécution

- Pile: zone mémoire marquée par un pointeur de sommet (dans un registre)
- A l'appel on empile:
  - le bloc des données de la procédure appelée,
  - l'adresse de retour dans la procédure appelante,
  - la base de ces informations est mémorisée dans un registre (ainsi que le sommet de la pile)
- Au retour,
  - on décrémente le sommet de pile de la taille du bloc donnée, sauf éventuellement le résultat
  - on fait un saut à l'adresse de retour qui était mémorisée.
- Permet de gérer les appels récursifs.

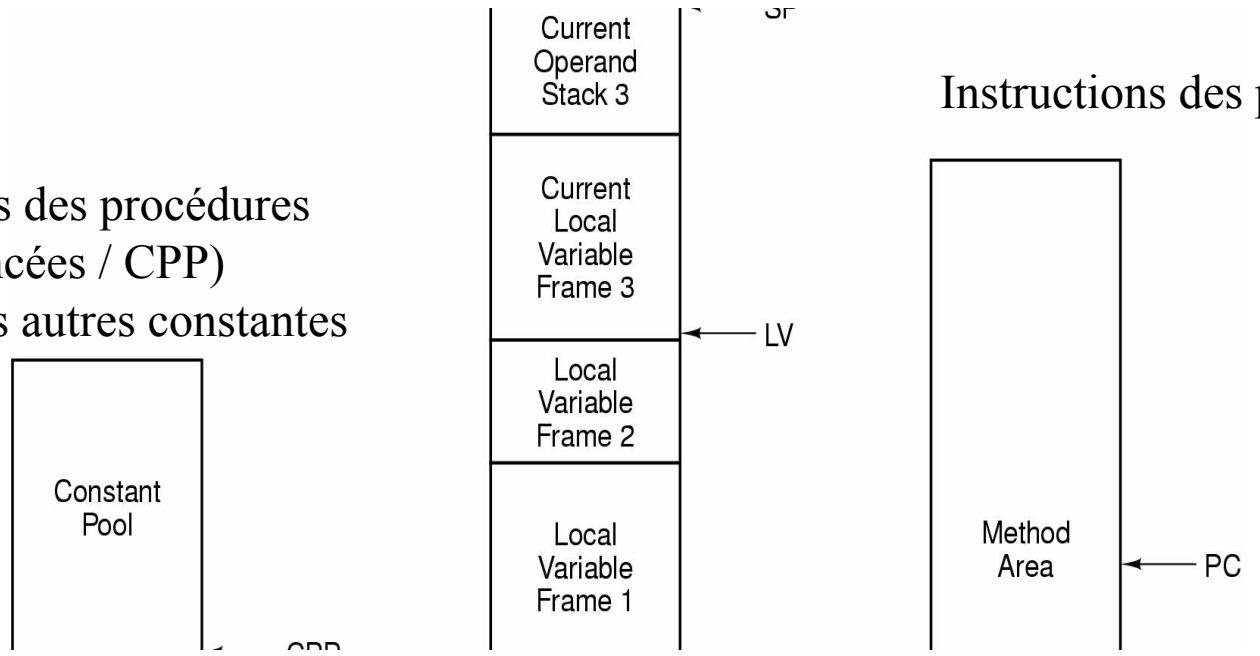
Une machine à pile sur les entiers: l'IJVM

Pile d'exécution+ opérations avec bloc de variables locales

- Modèle mémoire:

Instructions des procédures

adresses des procédures (référencées / CPP)
diverses autres constantes



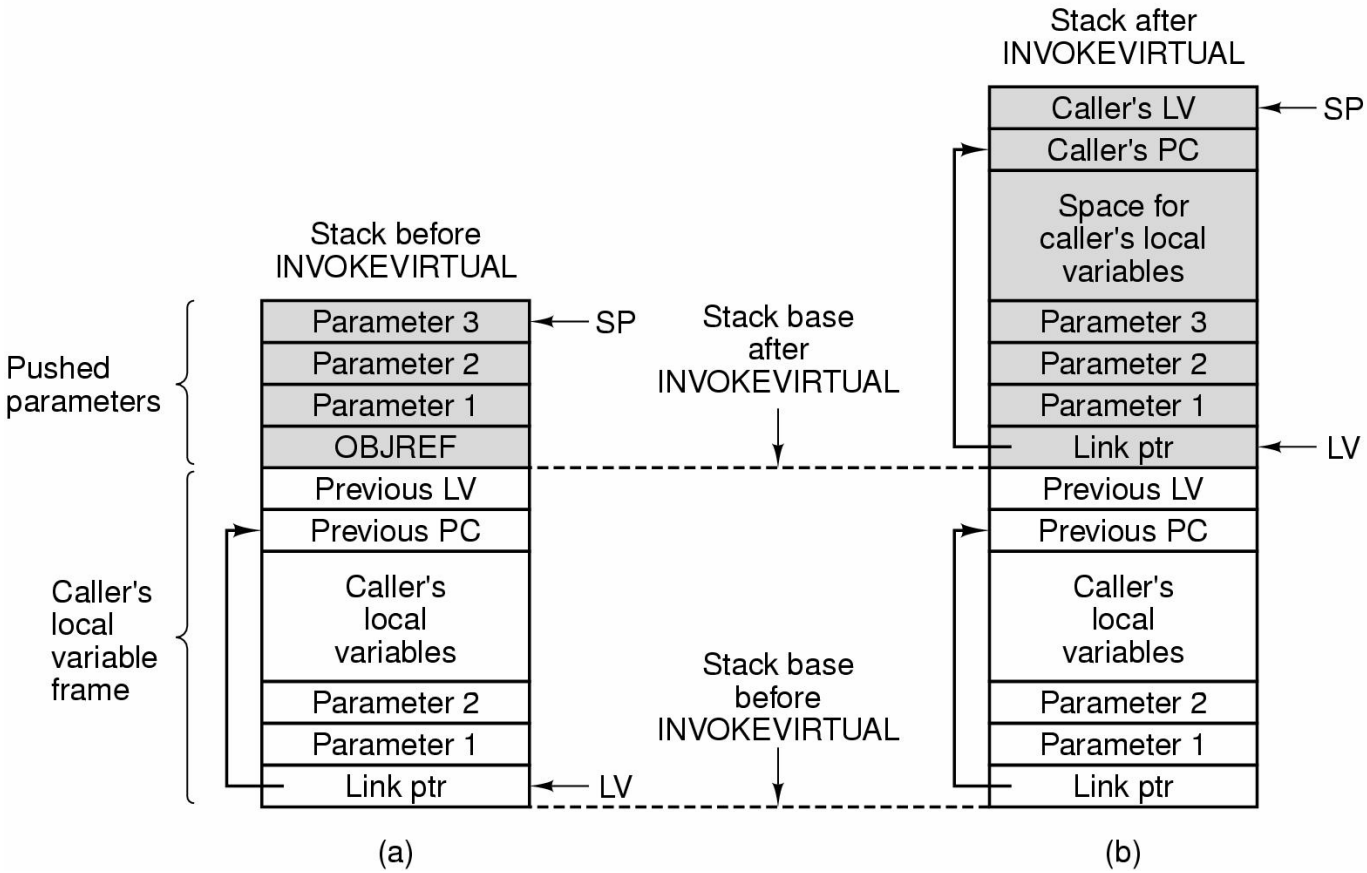
Organisation de la mémoire

- En octets pour les instructions
- En mots de 32 bits pour les données
- Mais toute donnée est accédée par le biais d'un index par rapport à l'une des bases (SP, LV, CPP)

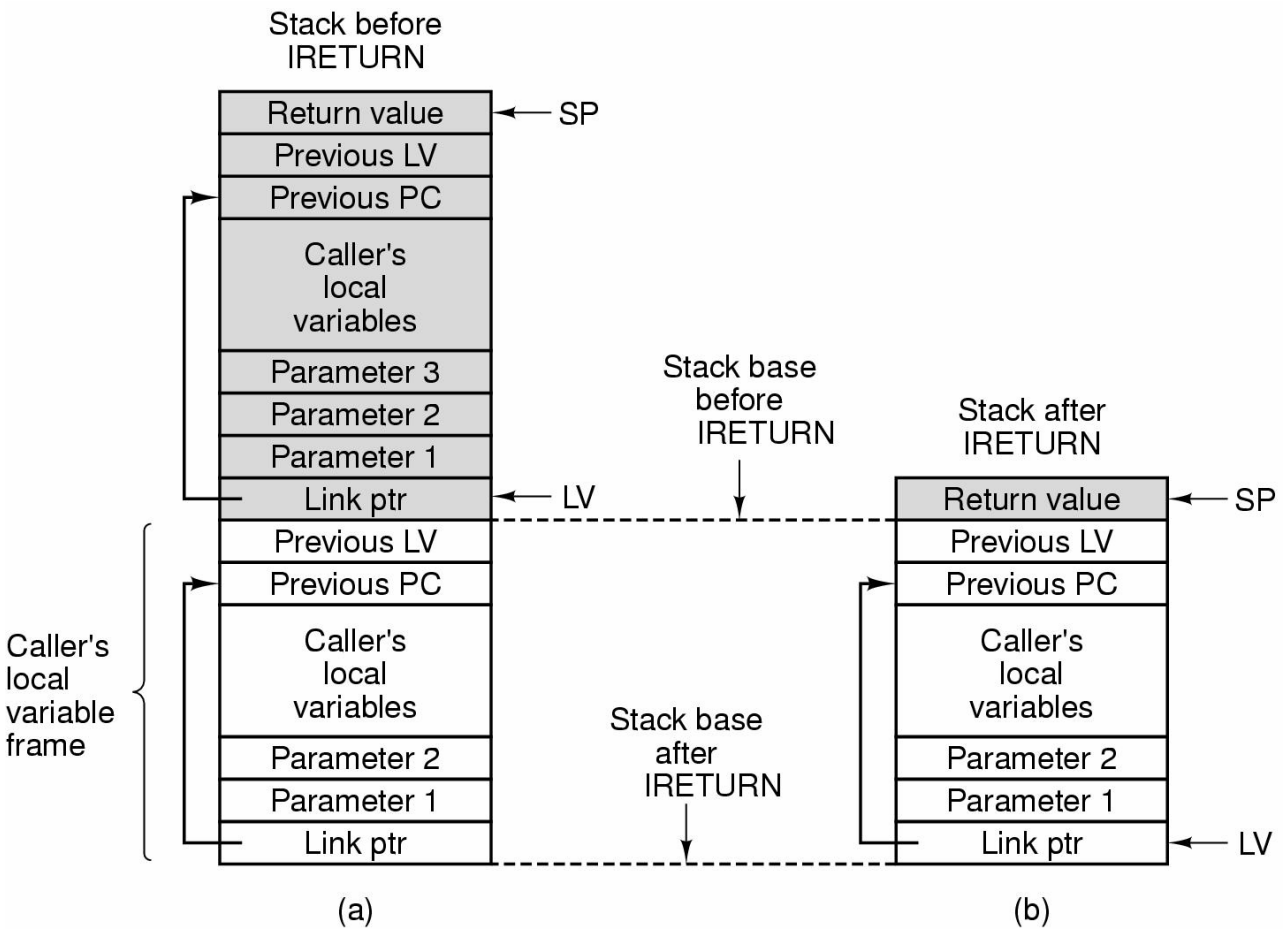
Instructions de l'IJVM

BIPUSH octet	push octet dans la pile	0x010
DUP	Duplique le mot du sommet et l'empile	0x59
GOTO offset	Branchement inconditionnel	0xA7
IADD	pop de deux mots, push de leur somme	0x60
IAND	pop de deux mots, push de leur ET	0x7E
IFLEQ offset	pop un mot de la pile, branchement s'il vaut 0	0x99
IFLT offset	pop un mot de la pile branchement s'il est <0	0x9B
IF_ICMPEQ offset	pop deux mots, branchement si égaux	0x9F
IINC numvar const	additionne constante à variable locale	0x84
ILOAD numvar	push une variable locale dans la pile	0x15
INVOKEVIRTUAL dep	invoque une méthode, dep est un index/ bloc de consantes	0xB6
IOR	pop de deux mots dans la pile, push de leur OR	0x80
IRETURN	retour de méthode avec valeur entière	0xAC
ISTORE numvar	pop mot de la pile et range dans variable locale	0x36
ISUB	pop deux mots, push différence	0x64
LDC_W index	push constante de zone constante	0x13
NOP	rien	0x00
POP	efface sommet pile	0x57
SWAP	permute deux mots du sommet de pile	0x5F
WIDE	préfixe d'instruction: l'instruction a un paramètre numvar ou const sur 16 bits (ordinairement sur 8)	0xC4

Appel de procédure



retour de procédure



Exemple

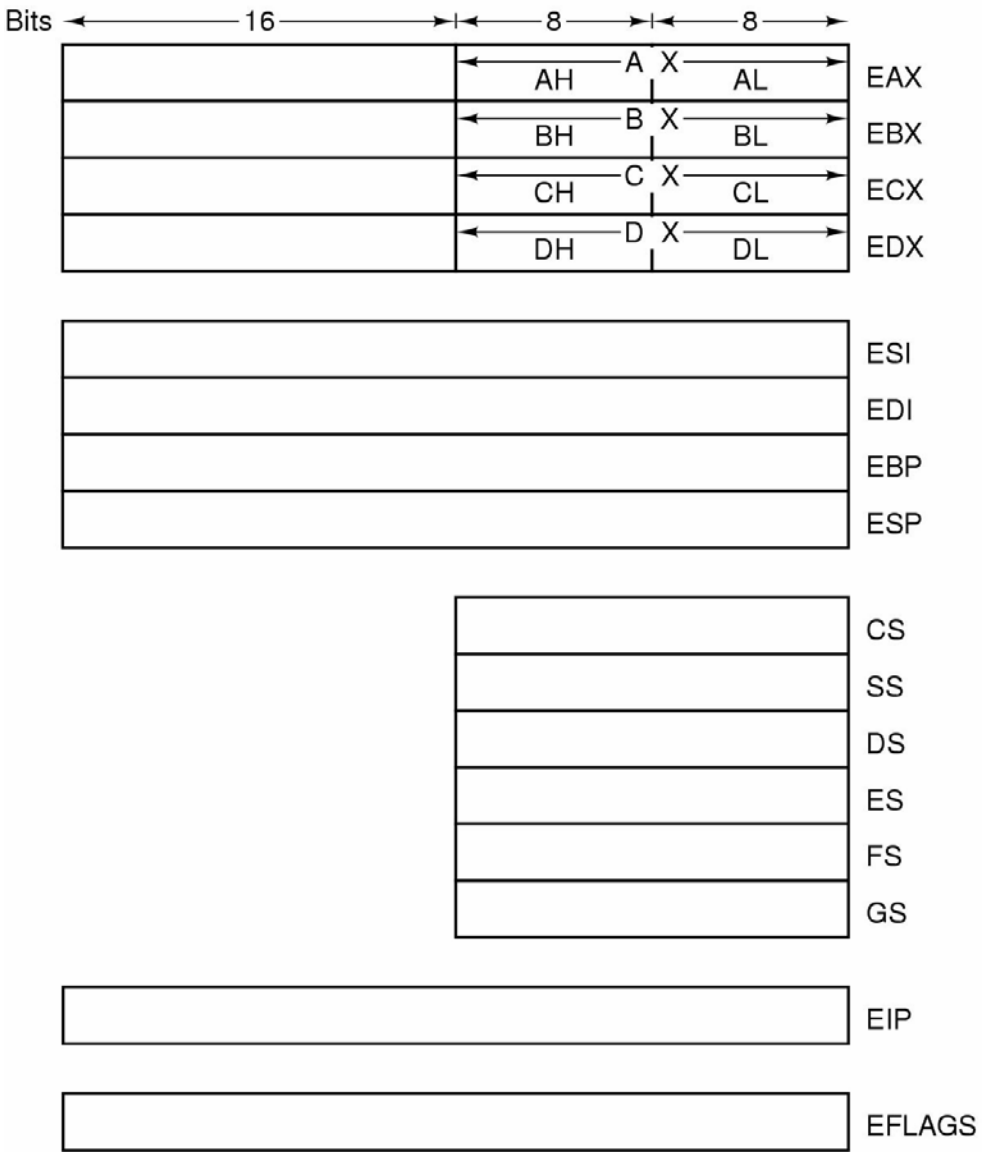
```
i=j+k;
if (i==3)
k=0;
else
j=j-1
```

i,j,k ont respectivement pour index 1,2,3 par rapport à LV

```
        ILOAD j          0x15 0x02
        ILOAD k          0x15 0x03
        IADD             0x60
        ISTORE i         0x36 0x01
        ILOAD i          0x15 0x01
        BIPUSH 3         0x10 0x03
        IF_ICMPEQ L1 0x9F 0x00 0x0D
        ILOAD j          0x15 0x02
        BIPUSH 1         0x10 0x01
        ISUB             0x64
        ISTORE j         0x36 0x02
        GOTO L2          0xA7 0x00 0x07
L1:     BIPUSH 0         0x10 0x00
        ISTORE k         0x36 0x03
L2:
```

Langage d'assemblage du Pentium II

- plusieurs modes opératoires (pour compatibilité commerciale)
- Mémoire découpée en segments puis en octets. Dans un segment, un octet est repéré par une adresse sur 32 bits.
- Les registres sont plus ou moins spécialisés et de taille différentes. chaque registre de 32 bits peut être vu comme 8 bits, 16 bits.
- instructions de taille variable, nombreus formats, nombreuses instructions -> décodage ifficile
- Beaucoup d'instructions font directement référence à la mémoire
- structure irrégulière des registres (plus difficile d'optimiser un compilateur)



Registres du pentium

Langage d'assemblage de l'UltraSparc

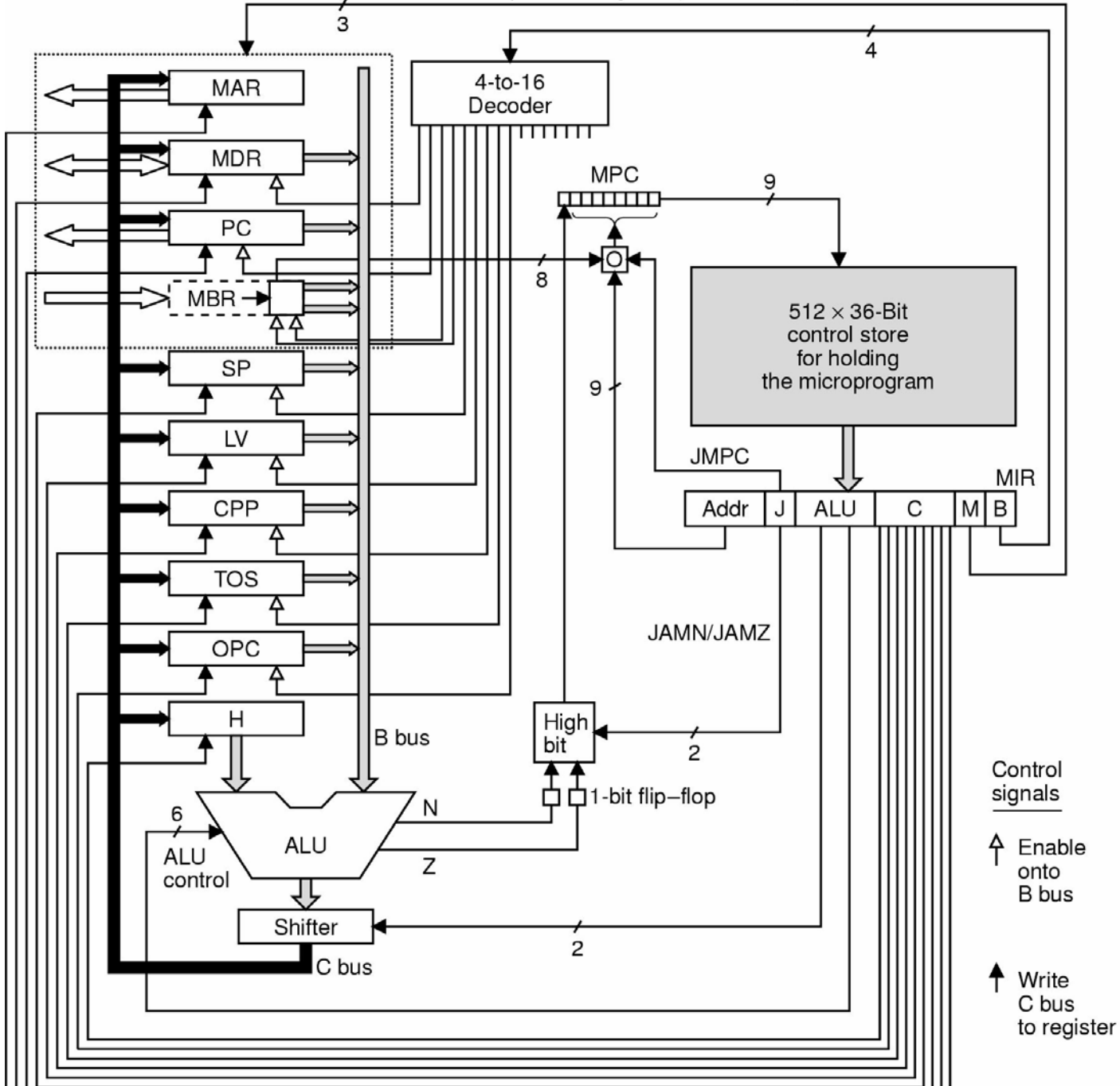
- Mémoire découpée en mots de 64 bits
- 32 Registres généraux de 64 bits avec des usages spécialisés courants pour les appels de procédure et 32 registres généraux pour les calculs flottants
- utilisation des registres pour simuler une pile pour les appels de procédure. Le système d'exploitation gère les recopies en mémoire lorsque le nombre de registres est insuffisant.
- instructions de taille fixe (32 bits) sauf une.
- Transferts mémoire-registres, les autres instructions travaillent sur les registres.
- Relativement peu d'instructions (RISC)

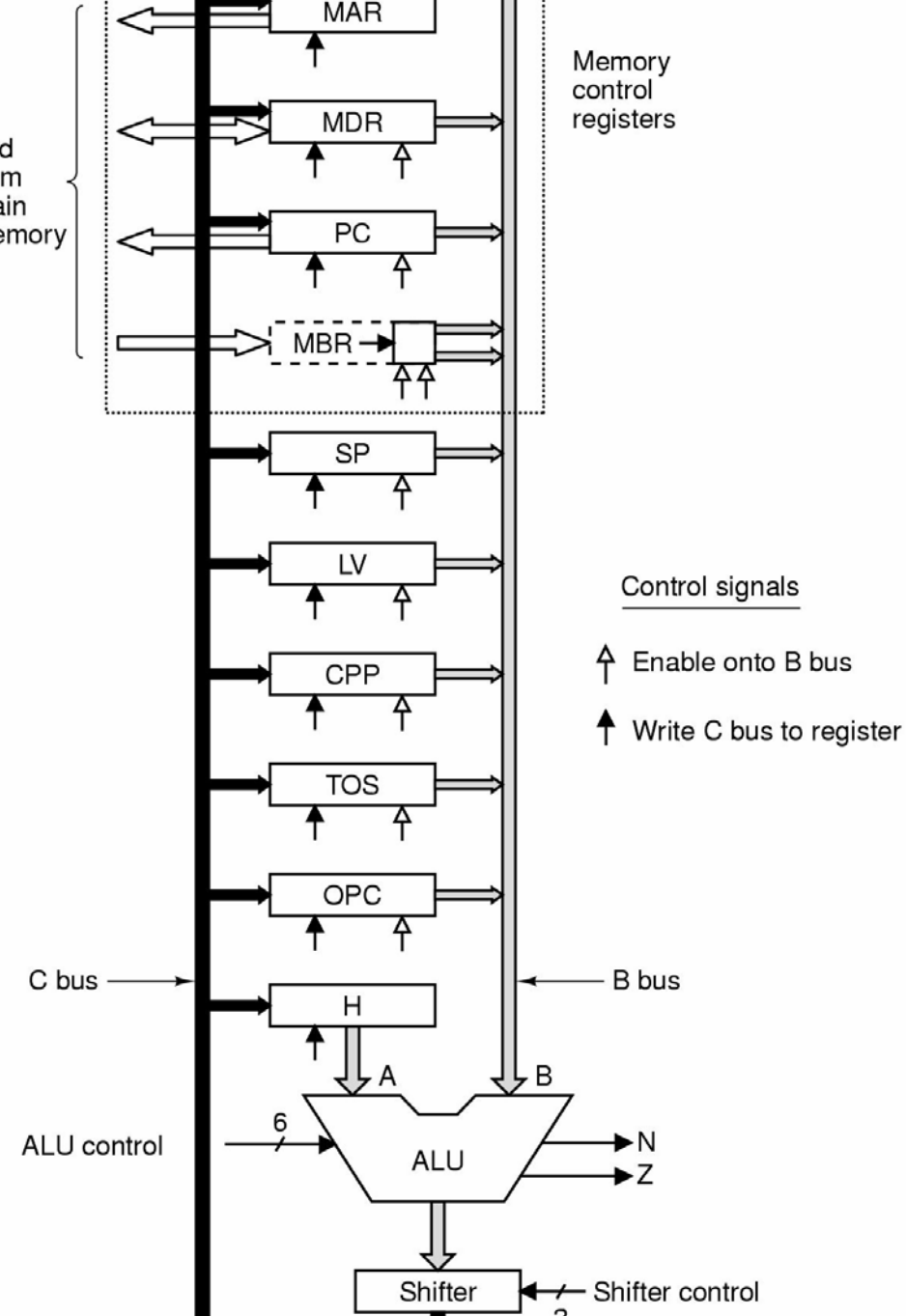
Couche micro-architecture

- Composants (registres, bus, signaux et circuits de commande, ALU)
- Micro-programme
  - micro-instruction: décrit les signaux de commande pour un cycle de l'architecture
  - une instruction de la couche ISA peut correspondre à une suite de micro-instructions
  - interpréteur.

Etude de MIC-1

- Une micro-architecture pour implémenter la machine IJVM.
- Caractéristique: ne travaille que sur des entiers (32 bits).


3
4
MAR
MDR
PC
MBR
SP
LV
CPP
TOS
OPC
H
4-to-16
Decoder
MPC
9
8
9
512 × 36-Bit
control store
for holding
the microprogram
JMPC
MIR
Addr
J
ALU
C
M
B
JAMN/JAMZ
High
bit
2
1-bit flip–flop
B bus
N
Z
6
ALU
ALU
control
Shifter
2
C bus
Control
signals
Enable
onto
B bus
Write
C bus
to register



Chemin des données

Description des fonctions des registres

- PC: compteur ordinal
- MBR : registre instruction (un octet)
- SP: pointeur de pile
- LV: base de la zone variable dans la pile
- MAR: registre d'adresse
- MDR: registre de données
- CPP: base du pool de constantes
- TOS; OPC registres complémentaires, utilisée pour la pile et comme bloc note.
- H: registre d'entrée/sortie de l'UAL
- N,Z : signaux de sortie de l'UAL (negatif, zéro)

type d'opération

F0	F1	ENA (valide entrée A)	ENB (valide entrée B)	INVA (inverse entrée A)	INC (force une retenue sur le bit de poids faible)	Fonction
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	Non A
1	0	1	1	0	0	Non B
1	1	1	1	0	0	A+B
1	1	1	1	0	1	A+B+1
1	1	1	0	0	1	A+1
1	1	0	1	0	1	B+1
1	1	1	1	1	1	B-A
1	1	0	1	1	1	B-1
1	1	1	0	1	1	-A
0	0	1	1	0	0	A et B
0	1	1	1	0	0	A ou B
0	1	0	0	0	0	0
0	1	0	0	0	1	1
0	1	0	0	1	0	-1

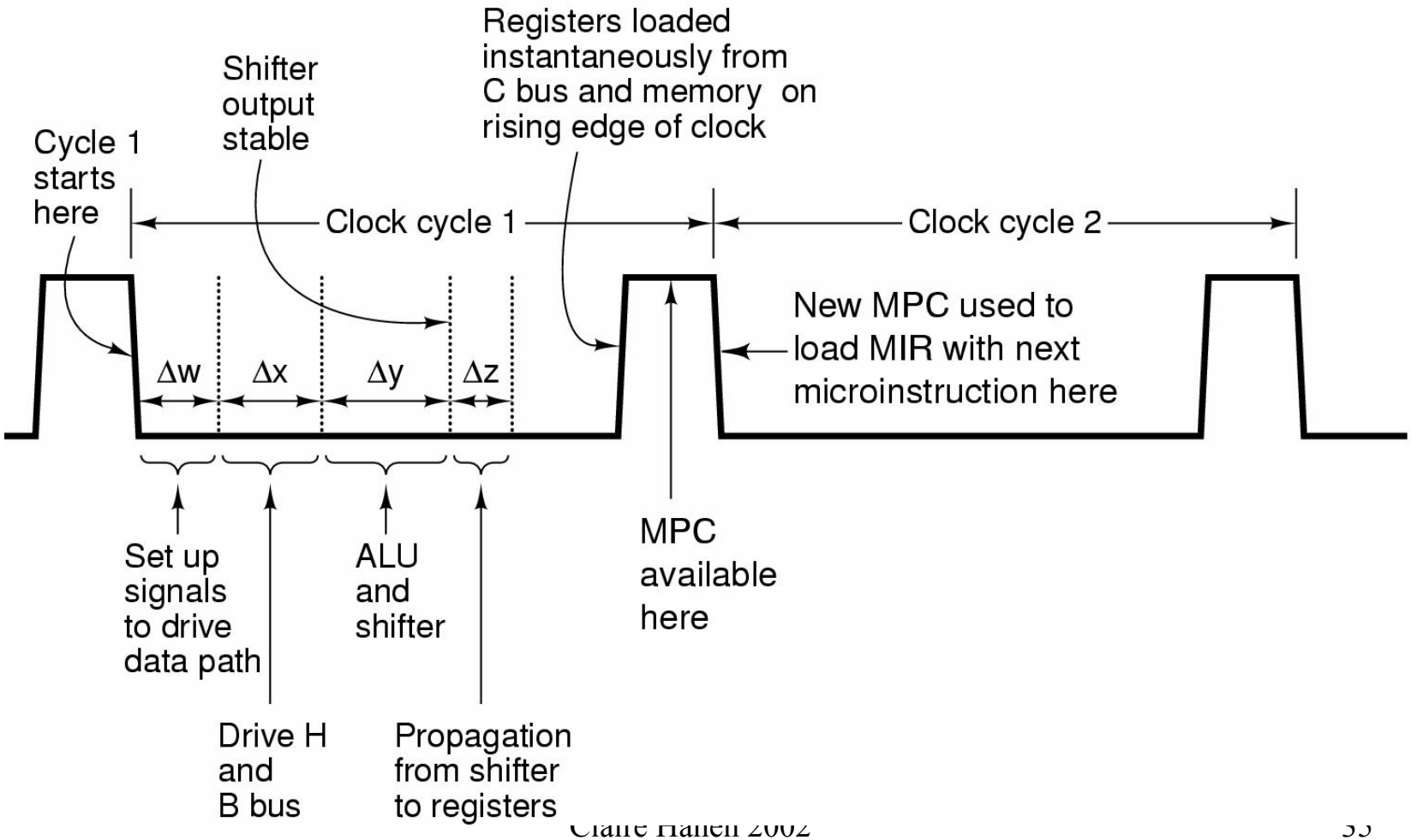
ALU

SLL8: décale à gauche d 8 bits sortie ALU
SRA1: décalage à droite d'1 bit sortie ALU
sans changer bit décalé

Décaleur

Chronologie du chemin

- On peut lire et écrire un registre pendant un même cycle.



Transferts avec la mémoire

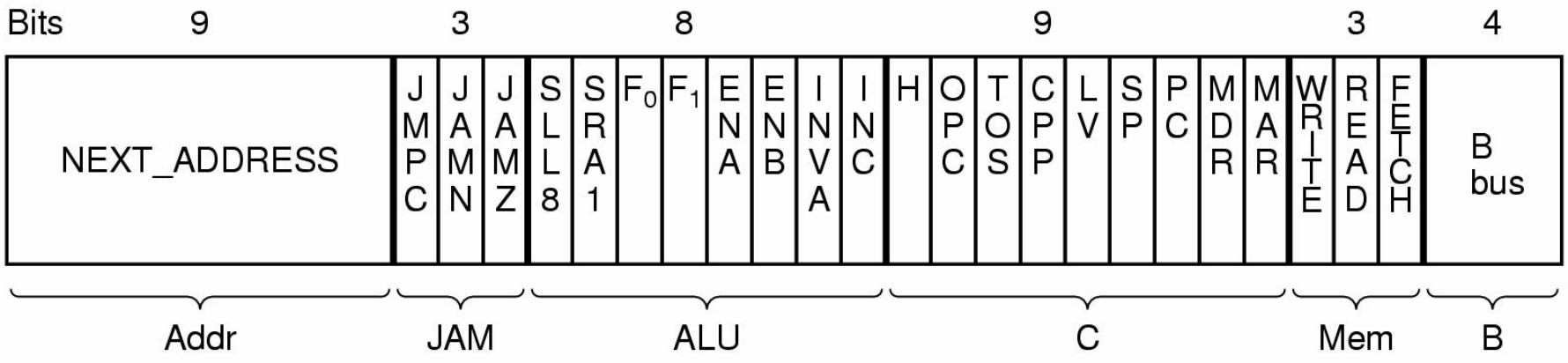
- port 32 bits (MAR/MDR) MAR contient le numéro du mot
- ports 8 bits (PC/MBR) PC contient le numéro de l'octet.
- MBR est complété pour faire un mot de 32 bits (sur le bus) avec ou sans extension de signe.
- registres commandés par signaux qui le connectent ou non au bus B/C
- signaux de lecture/écriture en mémoire et de recherche d'instruction (rd,wr,fetch)

Micro-instruction

- Signaux de commande nécessaires:
  - 9 pour ecriture bus C->registre
  - 9 pour copie regisre-> bus B
  - 8 UAL + décaleur
  - 3 signaux transferts mémoire
- Accès à la micro-instruction suivante

une microinstruction sur 36 bits

- les micro-instructions sont en mémoire (512 mots de 36 bits)
- introduction d'un décodeur pour l'entrée du bus B
- l'adresse de la micro-instruction suivante est directement mise dans l'instruction
- JAM: sert à faire des sauts éventuels dans le micro-programme.



B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9 -15 none

Mémoire de commande et micro-programme

- registre MPC qui contient l'adresse de la prochaine micro-instruction.
- registre MIR: contient la micro-instruction courante.
- Un cycle
  - –MIR chargé depuis la mémoire de commande à l'adresse MPC
  - –signaux propagés sur chemin de données
  - –données en entrée de l'UAL
  - –opération effectuée, N, Z activés
  - –sortie sur bus C puis chargement registres. Eventuellement changement MDR
  - –adresse suivante dans MPC
  - –Analyse champ JAM
  - –Si un ou plusieurs bits à 1, BPF= (JAMZ et Z) ou (JAMN et N) ou adresse suivante
  - –Modification de MPC