

Langage d'assemblage

Contexte

- ❑ Les CPU exécutent des instructions élémentaires (additions, recherche dans la mémoire, etc)
- ❑ Les instructions sont de la forme :
code_mnemo operande1, operande2, ...
- ❑ Il n'y a pas de variables mais des cases mémoires, on distingue :
 - Les cases mémoires dans le CPU appelé *registres* (bascules), noms commençant par \$ ou %
 - Les cases mémoires de la mémoire centrale (dont les adresses sont contenues dans un ou plusieurs registres)

Exemples d'instructions (MIPS)

$$(x-1)*3 + x*x$$

```
move    $v0, $a0 # a0 = x
li      $a1, 1
sub     $v0, $v0, $a1
li      $a1, 3
mul     $v0, $v0, $a1
move    $a1, $a0
move    $a2, $a0
mul     $a1, $a1, $a2
add     $v0, $v0, $a1
```

Jeu d'instructions : quel est l'ensemble des opcodes ?

- ❑ Un CPU exécute une suite d'instructions élémentaires. Jeu d'instructions = ensemble des instructions reconnues
- ❑ Plusieurs familles d'instructions :
 - Arithmétiques et logiques : `add`, `mul`, `sub`, `div`, `shl`, `shr`, `inc`, `dec` ...
 - Accès mémoire :
 - chargement : `load` (mem -> registre)
 - rangement : `store` (registre -> mem)
 - `mov` : divers modes
 -

Jeu d'instructions : quel est l'ensemble des codes op ? (suite)

- comparaison : `cmp`
- branchement : `call`, `jump`, `branch`
- accès registre contrôleur : `in`, `out`
- pile : `push`, `pop`
- divers : op. flottantes

□ Les op. arith. et log. positionnent généralement le registre des flags :

CF : carry flag (dernière retenue, n+1e bit)

ZF : zero flag

SF : sign flag

OF : overflow flag

Jeu d'instructions : quel est l'ensemble des codes op ? (suite)

- ❑ Les opérations de branchement s'appuient sur les tests des flags :

beq \$t1, \$t2, .L0 ZF = 1

slt \$t1, \$t2, \$t3 # if (\$t1 < \$t2) then \$t3=1
 else \$t3 = 0

\$t1-\$t2 et SF = 1

- ❑ ble, bge, blt, bgt, bnez, ...

Modes d'adressage : où se trouvent les opérandes ?

❑ Adressage registre :

add \$t1, \$t2, \$t3 # \$t1 = \$t2 + \$t3

❑ Adressage immédiat :

addi \$t1, \$t2, 18 # \$t1 = \$t2 + 18

❑ Adressage basé :

lw \$t1, 12(\$t2) # \$t1 = Memoire[\$t2 + 12]

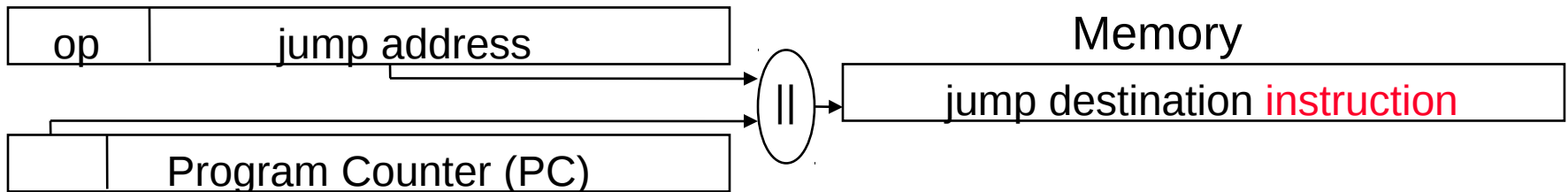
Modes d'adressage : où se trouvent les opérandes ? (suite)

❑ Adressage relatif au PC :

beq \$t1, \$t2, 8 # PC = PC+8 si \$t1 = \$t2

❑ Adressage pseudo-direct :

j 2000 # PC = concat(4 bits poids forts PC, 2000)



MIPS Instructions

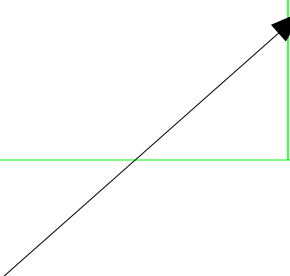
Category	Instr	OpC	Example	Meaning
Arithmetic (R & I format)	add	0 & 20	add \$s1, \$s2, \$s3	$\$s1 = \$s2 + \$s3$
	subtract	0 & 22	sub \$s1, \$s2, \$s3	$\$s1 = \$s2 - \$s3$
	add immediate	8	addi \$s1, \$s2, 4	$\$s1 = \$s2 + 4$
	shift left logical	0 & 00	sll \$s1, \$s2, 4	$\$s1 = \$s2 \ll 4$
	shift right logical	0 & 02	srl \$s1, \$s2, 4	$\$s1 = \$s2 \gg 4$ (fill with zeros)
	shift right arithmetic	0 & 03	sra \$s1, \$s2, 4	$\$s1 = \$s2 \gg 4$ (fill with sign bit)
	and	0 & 24	and \$s1, \$s2, \$s3	$\$s1 = \$s2 \& \$s3$
	or	0 & 25	or \$s1, \$s2, \$s3	$\$s1 = \$s2 \mid \$s3$
	nor	0 & 27	nor \$s1, \$s2, \$s3	$\$s1 = \text{not } (\$s2 \mid \$s3)$
	and immediate	c	and \$s1, \$s2, ff00	$\$s1 = \$s2 \& 0\text{xff}00$
	or immediate	d	or \$s1, \$s2, ff00	$\$s1 = \$s2 \mid 0\text{xff}00$
	load upper immediate	f	lui \$s1, 0xffff	$\$s1 = 0\text{xffff}0000$

MIPS Instructions

Category	Instr	OpC	Example	Meaning
Data transfer (I format) Cond. branch (I & R format) Uncond. jump	load word	23	lw \$s1, 100(\$s2)	\$s1 = Memory(\$s2+100)
	store word	2b	sw \$s1, 100(\$s2)	Memory(\$s2+100) = \$s1
	load byte	20	lb \$s1, 101(\$s2)	\$s1 = Memory(\$s2+101)
	store byte	28	sb \$s1, 101(\$s2)	Memory(\$s2+101) = \$s1
	load half	21	lh \$s1, 101(\$s2)	\$s1 = Memory(\$s2+102)
	store half	29	sh \$s1, 101(\$s2)	Memory(\$s2+102) = \$s1
	br on equal	4	beq \$s1, \$s2, L	if (\$s1==\$s2) go to L
	br on not equal	5	bne \$s1, \$s2, L	if (\$s1 !=\$s2) go to L
	set on less than immediate	a	slti \$s1, \$s2, 100	if (\$s2<100) \$s1=1; else \$s1=0
	set on less than	0 & 2a	slt \$s1, \$s2, \$s3	if (\$s2<\$s3) \$s1=1; else \$s1=0
	jump	2	j 2500	go to 10000
	jump register	0 & 08	jr \$t1	go to \$t1
	jump and link	3	jal 2500	go to 10000; \$ra=PC+4

Conditionnelle MIPS

<pre>if (5 > 6) { a = a+1; } else { a = a + 2; }</pre>	<pre>li \$t0, 5 # \$t0 <- 5 li \$t1, 6 ble \$t0, \$t1, Else addi \$t2, \$t2, 1 # a dans \$t2 j Next Else : addi \$t2, \$t2, 2 Next : ...</pre>
---	---



étiquette (remplacée par une adresse lors de la compilation)

Itération MIPS

```
for(i=0; i<10;i++);
```

```
li $t0, 0    # i = 0  
li $t1, 10  
Loop: bge $t0, $t1, Next  
      addi $t0, $t0, 1  
      j Loop  
Next :...
```

Programme MIPS

- ❑ Contient deux sections :
 - Les instructions : section .text
 - Les variables globales : section .data

- ❑ Exemple : min max d'un tableau

Format des instructions

❑ Encodage des instructions, 3 formats :

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
OP	rs	rt	rd	shamt	funct	R format
OP	rs	rt	16 bit number			I format
OP	26 bit jump target					J format

❑ Exemple : encodage de **add \$s1, \$s2, \$s3**

R-type instruction Arith/Log/Shift/Comparaison

- Opcode : 0 (UAL reg), rd=17, rs=18, rt=19, funct=100000
- 000000 10010 10011 1001 00000 100000
- codage hexa : 0x02539020

Format des instructions

❑ Encodage des instructions, 3 formats :

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
OP	rs	rt	rd	shamt	funct	R format
OP	rs	rt	16 bit number			I format
OP	26 bit jump target					J format

❑ Exemple : encodage de **sw \$2, 128(\$3)**

- I-type memory address instruction
- Opcode : 101011, rs=00011, rt=00010, imm=0000000010000000
- 101011 00011 00010 0000000010000000

Format des instructions

❑ Encodage des instructions, 3 formats :

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	
OP	rs	rt	rd	shamt	funct	R format
OP	rs	rt	16 bit number			I format
OP	26 bit jump target					J format

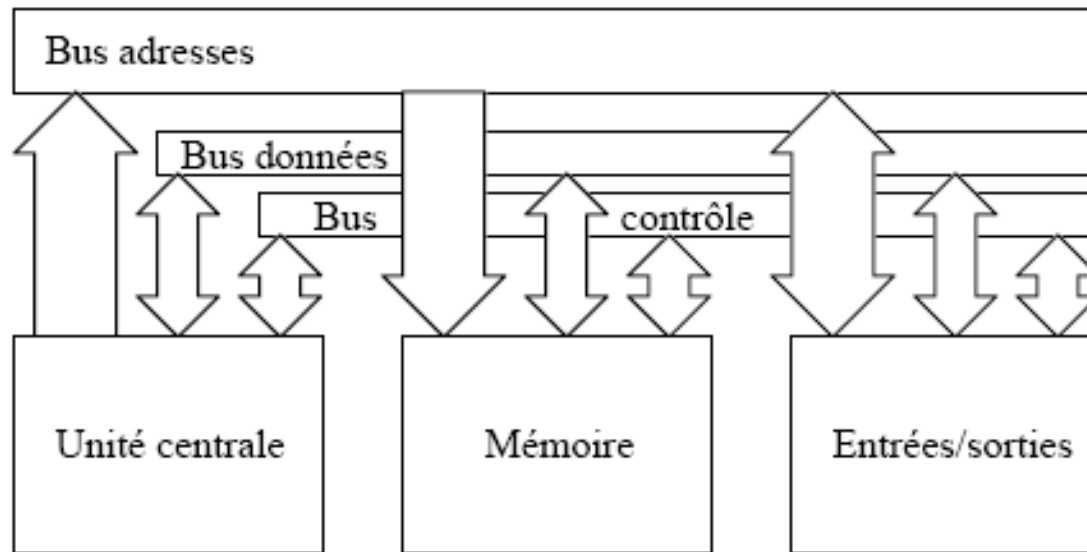
❑ Exemple : encodage de **J 128**

- J-type pseudodirect jump instruction
- Opcode : 000010, 26-bit pseudodirect address is $128/4 = 32$
- 000010 00000000000000000000100000

Le CPU : un circuit séquentiel

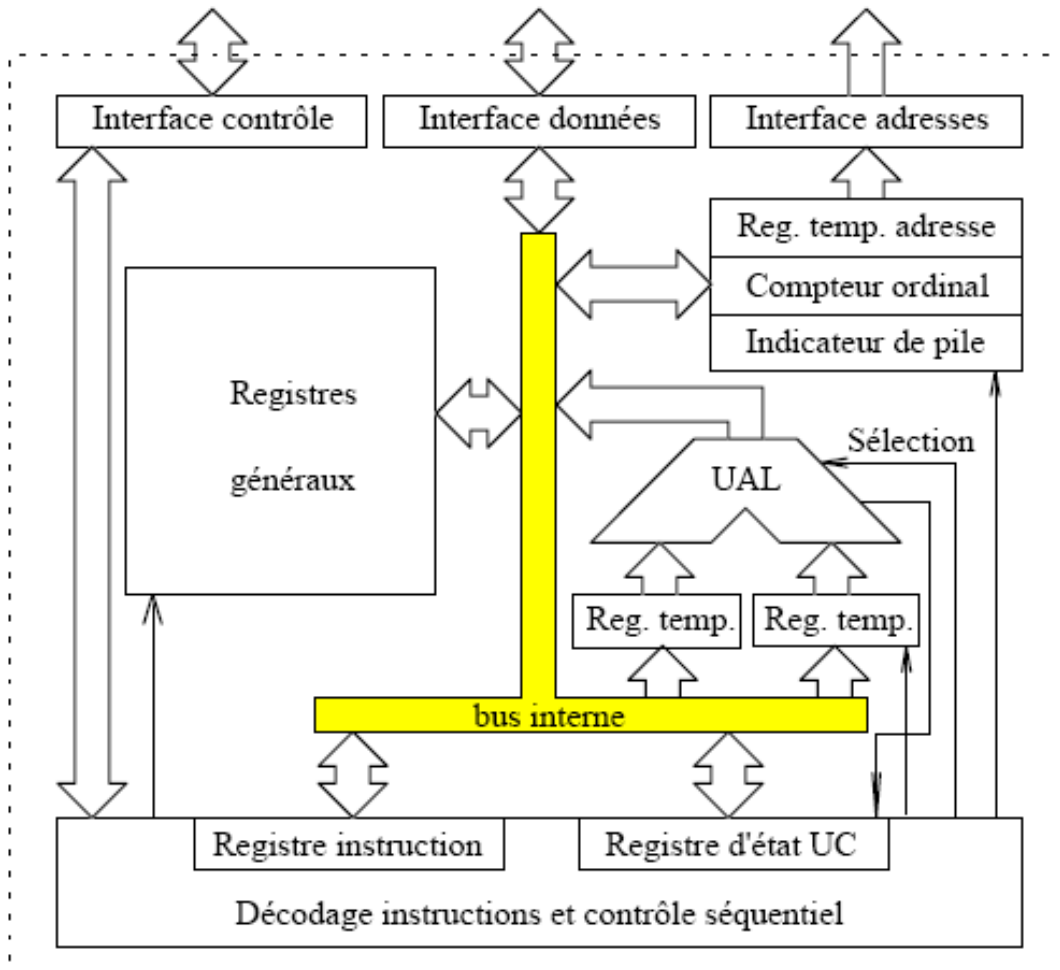
Architecture d'une machine

- ❑ Le processeur (CPU) est un circuit séquentiel relié à une mémoire vive, à une mémoire morte et aux périphériques



- ❑ Le circuit est réalisé après la définition d'un jeu d'instructions

Vision simplifiée d'un CPU

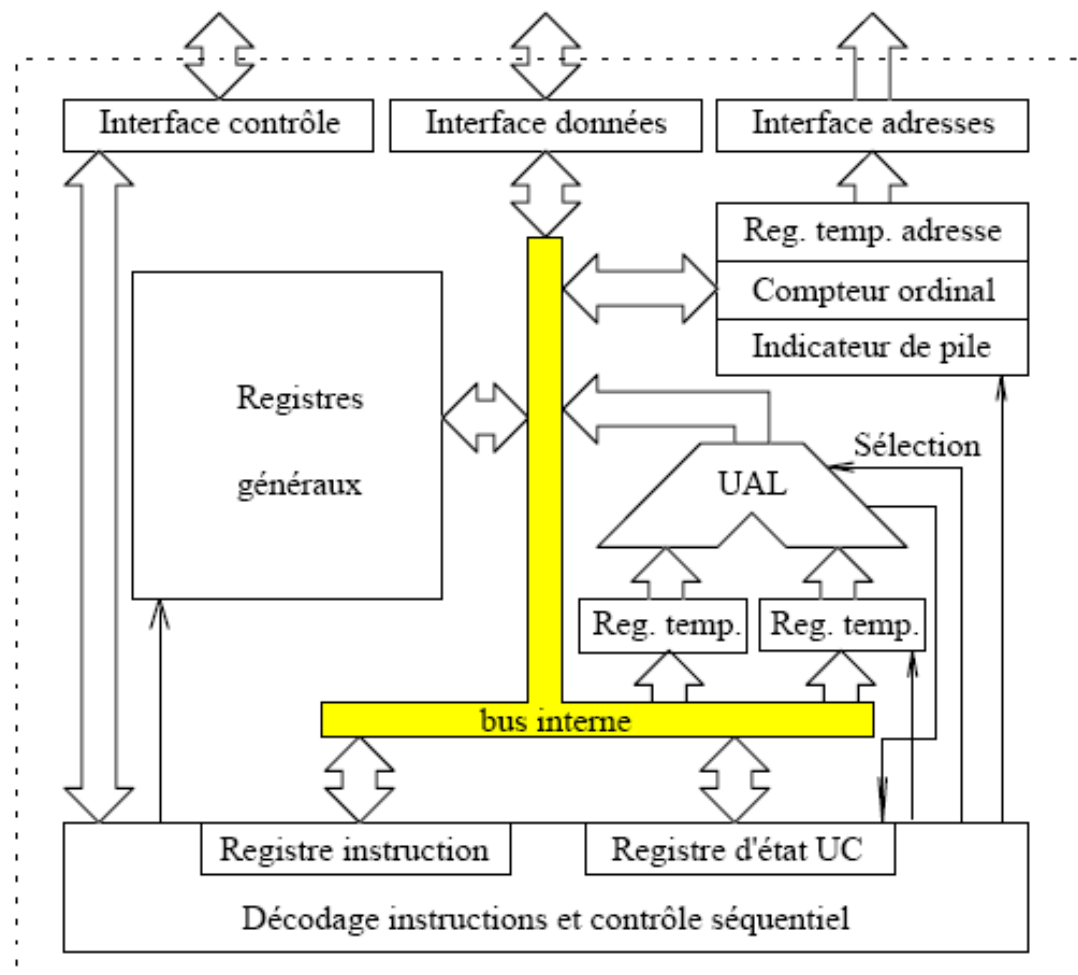


- registres généraux
- registres spéciaux :
 - SP : Stack Pointer
 - PC : Program Counter (ou CO)
 - IR : Instruction Register
 - Flags

SP : pour gérer les appels
de fonctions

Flags : overflow, zero,
carry, ...

Vision simplifiée d'un CPU



Fonctionnement d'un CPU :

1. **Recherche** d'instruction : $IR = Mem[PC]$
2. **Décodage** : à partir de cette instruction, générer les signaux pour les unités (UAL, banc de reg)
3. **Lecture** des données
4. **Exécution**
5. **Écrire** les résultats
 $PC = PC + \text{résultat calcul}$