ARCHITECTURE DES ORDINATEURS

2ème Année Informatique

Chapitre 4:
Le processeur

Centre universitaire Mila
2022-2023

1

Microprocesseur

Un microprocesseur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme.
Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution. Il doit aussi prendre en compte les informations extérieures au système et assurer leur traitement. C'est le cerveau du système.

Microprocesseur

Un processeur est défini par:

- ✓ Son architecture, c'est-à-dire son comportement vu par le programmeur, liée à:
- ➢ Son jeu d'instructions(en anglais Instruction Set Architecture, ISA);
- ➢ la largeur de ses registres internes de manipulation de données(8, 16, 32, 64, 128)bits et leur utilisation;
- ➢ les spécifications des entrées–sorties, de l'accès à la mémoire, etc.

3

Microprocesseur

Un processeur est défini par:

- ✓ Ses caractéristiques, variables même entre processeurs compatibles,
- ➢ la cadence de son horloge exprimée en mégahertz (MHz) ou gigahertz (GHz);
- ➢ sa finesse de gravure exprimée en nanomètres (nm);
- ➢ son nombre de cœurs de calcul.

4

Architectures CISC/RISC

Actuellement l'architecture des microprocesseurs se composent de deux grandes familles :

- **L' architecture CISC** (Complex Instruction Set Computer)

Leurs instructions, de taille variable, sont variées et réalisent souvent des transferts avec la mémoire. Ces processeurs possèdent en général peu de registres, dont certains sont réservés pour des usages spécifiques.

Exemples: Intel 8086 et Motorola 68000 (pré-1985).

- **L'architecture RISC** (Reduced Instruction Set Computer)

Leurs instructions, de taille fixe, sont régulières et peu d'entre elles lisent ou écrivent en mémoire. Ces processeurs possèdent en général de nombreux registres, lesquels sont uniformes.

Exemples: Alpha, Sparc, Mips, PowerPC (post-1985).

- Le Pentium d'Intel mélange les deux designs...

5

Architectures CISC/RISC:

Architecture RISC	Architecture CISC
instructions simples ne prenant qu'un seul cycle	instructions complexes prenant plusieurs cycles
instructions au format fixe	instructions au format variable
décodeur simple (câblé)	décodeur complexe (microcode)
beaucoup de registres	peu de registres
peu de modes d'adressage	beaucoup de modes d'adressage
compilateur complexe	compilateur simple

6

Mode Utilisateur/Superviseur:

Afin de mettre en oeuvre les mécanismes de protection nécessaires pour un système, le processeur possède deux modes de fonctionnement :

- **Le mode superviseur:** permet une plus grande sécurité pour le système. Quand le processeur est en mode superviseur, l'utilisateur ne peut pas accéder à certaines zones (ex: mémoire, registres...)
- **Le mode utilisateur:** Ce mode permet à l'utilisateur d'accéder uniquement aux zones réservées aux utilisateurs. Le processeur part en exception si l'utilisateur accède à une zone privilégiée.

Note: quand le processeur est en mode superviseur, le système peut accéder aux zone protégées et non protégées.

7

Le microprocesseur MIPS R3000

- MIPS (de l'anglais Microprocessor without interlocked pipeline stages) a été développée par la compagnie MIPS Computer Systems Inc., basée en Californie.
- En 1988, la société MIPS Computer Systems, présente le MIPS R3000 qui succède au MIPS R2000.
- Son jeu d'instructions est de type RISC.
- Le processeur MIPS R3000 est un processeur 32 bits.
- Le principal marché du MIPS: les applications embarquées, les ordinateurs de poche, les routeurs Cisco et les consoles de jeux vidéo (Nintendo 64 et Sony PlayStation, et PSP)…
- Il existe plusieurs réalisations industrielles de cette architecture (Siemens, Toshiba, Philips, Silicon Graphics,etc...)

8

Les registres

Les registres sont des zones de stockage temporaire située dans le processeur.

- La valeur de certains registres peut être lue ou modifiée par les instructions.
- Selon les modes de fonctionnement du processeur, il existe deux catégories de registres:
  - Les registres non protégés (Accessibles en mode utilisateur)
  - Les registres protégés (Accessibles en mode superviseur)

9

Les registres généraux

- La machine MIPS dispose de 32 registres d'usage général de 32 bits chacun, dénotés $0 à $31.
- Les registres peuvent également être identités par un mnémonique indiquant leur usage conventionnel.

10

Les registres généraux

Registre	Numéro	Usage	Préservé par l'appelé
zero	0	Constante 0	
at	1	Réservé pour l'assembleur	N
v0	2	Évaluation d'expressions et	N
v1	3	résultats d'une fonction	N
a0-a3	4-7	Arguments 1 à 4	N
t0-t7	8-15	Temporaires	N
s0-s7	16-23	Temporaires sauvegardés	O
t8	24	Temporaire	N
t9	25	Temporaire	N
k0	26	Réservé pour le système d'exploitation	N
k1	27	Réservé pour le système d'exploitation	N
gp	28	Pointeur vers la zone des variables globale	O
sp	29	Pointeur de pile (stack pointer)	O
fp	30	Pointeur de bloc (frame pointer)	O
ra	31	Adresse de retour (pour les fonctions)	O

11

Les registres spéciaux

-**PC** Registre compteur de programme (Program Counter), qui contient l'adresse mémoire de la prochaine instruction.
-**Hi** et **Lo** : registres utilisés pour la multiplication et la division
- **SR** (status): Status Register :c'est le registre d'état. Il contient les masques d'interruption et le mode
-**CAUSE** : contient la cause de l'interruption / exception
-**EPC** : Exception PC (Program Counter) : adresse de l'instruction qui a causé l'exception
- **BAR**(vaddr) : Bad Address Register
ce registre contient l'adresse fautive en cas d'exception de type « adresse illégale »

12

Format d'une instruction

Il y a 3 formats d'instructions différents :

- R : Register to Register – Register Type
- I : Accès mémoire, branchements – Immediate Type
- J : Instructions de sauts (non-conditionnels) : Jump Type

13

Format d'une instruction

- **Le format R** : est utilisé par les instructions nécessitant deux registres sources (RS et RT) et un registre résultat (RD).
- **Le format I**: est utilisé par les instructions de lecture/écriture mémoire, par les instructions utilisant un opérande immédiat, ainsi que par les branchements conditionnels.
- **Le format J**: n'est utilisé que pour les branchements inconditionnels.

14

Format d'une instruction

- **Le format R**

Opcode	Rs	Rt	Rd	Sham	Func
6	5	5	5	5	6

- **Le format I**

Opcode	Rs	Rt	Immed
6	5	5	16

- **Le format J**

Opcode	Immed
6	26

15

Format d'une instruction

- le code binaire **opcode** (operation code) identifiant l'instruction Sur 6 bits, il ne permet de coder que 64 instructions, ce qui même pour un processeur RISC est peu. Par conséquent, un champ additionnel function de 6 bits est utilisé pour identifier les instructions R-type.
- **Rd** est le registre destination (valeur sur 5 bits, donc comprise entre 0 et 31, codant le numéro du registre)
- **Rs** est le premier argument source
- **Rt** est le second argument source
- **Sham** (shift amount) est le nombre de bits de décalage, pour les instructions de décalage.
- **Func** 6 bits additionnels pour le code des instructions R-type, en plus du champ opcode.

16

pseudo-instruction

- pseudo-instruction (macro-instruction) est une instruction qui ne fait pas partie du jeu d'instructions machine, mais qui est acceptée par l'assembleur qui la traduit en une séquence d'instructions machine.
- Les pseudo-instructions utilisent le registre **$1** si elles ont besoin de faire un calcul intermédiaire. Il faut donc éviter d'utiliser ce registre dans les programmes.

17

pseudo-instruction

Pseudo-instruction move

Comment traduire A=B ?

- Sachant que A=$t0, B=$t1 et que le registre $0 vaut toujours 0 on peut écrire :
- add $t0, $0, $t1
- Il vaut mieux utiliser l'instruction move :
- move est une pseudo-instruction : sa traduction en langage machine
- est celle de add $t0, $0, $t1.

18

Codage des instructions

Le codage des instructions est principalement défini par les 6 bits du champ code opération de l'instruction (INS 31:26). Cependant, trois valeurs particulières de ce champ définissent en fait une famille d'instructions : il faut alors analyser d'autres bits de l'instruction pour décoder l'instruction. Ces codes particuliers sont :

- SPECIAL (valeur "000000"),
- BCOND (valeur "000001")
- COPRO (valeur "010000")

19

Codage des instructions

DECODAGE OPCOD

	000	001	010	011	100	101	110	111
000	SPECIAL	BCOND	J	JAL	BEQ	BNE	BLEZ	BGTZ
001	ADDI	ADDIU	SLTI	SLTIU	ANDI	ORI	XORI	LUI
010	COPRO							
011								
100	LB	LH		LW	LBU	LHU		
101	SB	SH		SW				
110								
111								

20

Codage des instructions

OPCOD = SPECIAL

	000	001	010	011	100	101	110	111
000	SLL		SRL	SRA	SLLV		SRLV	SRAV
001	JR	JALR			SYSCALL	BREAK		
010	MFHI	MTHI	MFLO	MTLO				
011	MULT	MULTU	DIV	DIVU				
100	ADD	ADDU	SUB	SUBU	AND	OR	XOR	NOR
101			SLT	SLTU				
110								
111								

21

Codage des instructions

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c011001	lui $1,4097	4: lw $t1, x
	0x00400004	0x8c290000	lw $9,0($1)	
	0x00400008	0x20010001	addi $1,$0,1	5: subi $t2,$t1,1
	0x0040000c	0x01215022	sub $10,$9,$1	
	0x00400010	0x3c011001	lui $1,4097	6: sw $t2, x
	0x00400014	0xac2a0000	sw $10,0($1)	

22

Codage des instructions

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c010001	lui $1,1	4: lui $1,1
	0x00400004	0x24090005	addiu $9,$0,5	5: li $t1,5

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x24090005	addiu $9,$0,5	4: li $t1,5
	0x00400004	0x240a0006	addiu $10,$0,6	5: li $t2,6
	0x00400008	0x012a5820	add $11,$9,$10	6: add $t3,$t1,$t2

23

Codage des adresses

- Les adresses dans les instructions ne sont pas sur 32 bits !
- Pour les instructions de type I : 16 bits
- =) Adresse = PC + signé(16 bits) * 4 **adressage relatif**
- Pour les instructions de type J : 26 bits

=) On obtient l'adresse d'un mot mémoire (de 32 bits) en ajoutant devant les 26 bits les 4 bits de poids fort de PC (Il faut multiplier par 4 pour l'adresse d'un octet)

adressage direct restreint

24

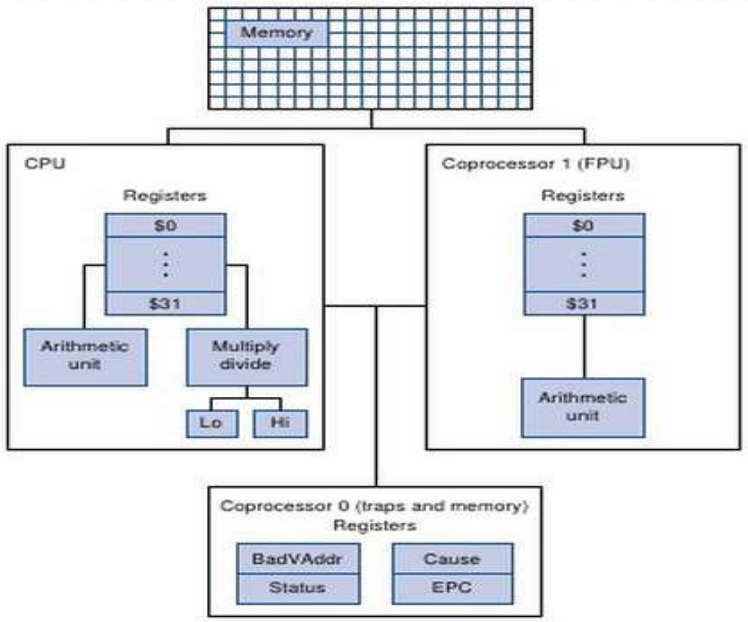
Co-processeurs de MIPS

- Le microprocesseur Mips travaille avec des coprocesseurs pour effectuer des opérations spécifiques.
- Le co-processeur 0: (processeur de contrôle système) gère le système de mémoire virtuelle, les exceptions ainsi que les transitions entre les modes Superviseur et Utilisateur.
- Le co-processeur FPU: (Floating Point Unit) Est chargé d'effectuer les opérations arithmétiques à virgules flottantes.

Il possède 32 registres nommés: $f0 – $f31

25

Co-processeurs de MIPS



26

ORGANISATION DE LA MÉMOIRE EN MIPS R3000

27

Mémoire et programme

- Quand on demande l'exécution d'un programme, celui-ci est chargé par le système d'exploitation (à partir du disque dur) dans une zone de la mémoire principale RAM.
- Chaque instruction est stockée dans une position mémoire.
- l'adresse (le numéro) de la case mémoire de début est transférée au processeur (placée dans le PC) pour qu'il commence l'exécution au bon endroit.

28

Segmentation mémoire du MIPS R3000

- Dans l'architecture MIPS R3000, la mémoire est divisée en deux segments:
- Le segment utilisateur, et le segment noyau (mode superviseur).
- Quand le processeur est en mode superviseur, les 2 segments sont accessibles.
- Quand le processeur est en mode utilisateur, seul le segment utilisateur est accessible.
- Le processeur part en exception si une instruction essaie d'accéder à la mémoire avec une adresse correspondant au segment système alors que le processeur est en mode utilisateur.

29

Segmentation mémoire du MIPS R3000

Un programme utilisateur utilise généralement trois sous segments(appelés sections) dans le segment utilisateur:

•**La section text** contient le code exécutable en mode utilisateur.

•**La section data** contient les données globales manipulées par le programme utilisateur.

•**La section stack** contient la pile d'exécution du programme utilisateur.

Trois sections sont également définies dans le segment noyau:

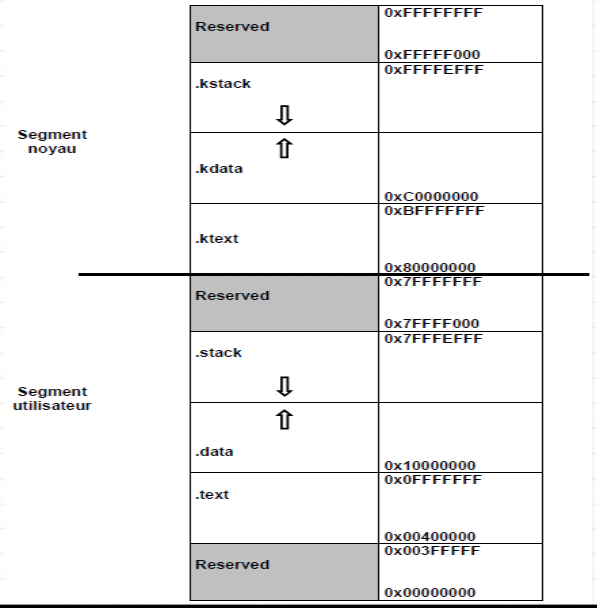
•**La section ktext** contient le code exécutable en mode superviseur.

•**La section kdata** contient les données globales manipulées par le système en mode superviseur.

•**La section kstack** contient la pile d'exécution du noyau.

30

Segmentation mémoire du MIPS R3000



Adressage mémoire:

La mémoire est vue comme un tableau d'octets, qui contient aussi bien les données que les instructions.

- L'unité d'adressage est l'octet.
- Les instructions sont codées sur 32bits.
- Pour enregistrer une instruction quatre(4)octets sont nécessaires.
- La représentation d'un mot(32-bits) en mémoire est comme suit:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Poids fort																								Poids faible							

Adressage mémoire:

Par exemple:

- num1: .word 42
- num2: .word 5.000.000
- 42_{10} est représentée en hexadécimal par:0x0000002A
- $5{,}000{,}000_{10}$ est représenté en hexadécimal par 0x004C4B40.
- La représentation de ces deux mots en mémoire est:

33

Adressage mémoire:

Variables	Valeur	Adresses
	?	0x100100C
	00	0x100100B
	4C	0x100100A
	4B	0x1001009
Num2 →	40	0x1001008
	00	0x1001007
	00	0x1001006
	00	0x1001005
Num1 →	2A	0x1001004
	?	0x1001003

34