ARCHITECTURE DES ORDINATEURS
Examen Décembre 2005
3H – Tous documents autorisés

EXECUTION D'INSTRUCTIONS

On utilise le jeu d'instructions MIPS décrit en annexe.

Q 1 : Donner l'instruction ou la suite des instructions MIPS pour effectuer les actions suivantes :

a) Mettre à zéro le registre R1
b) Mettre 00008000_H dans le registre R2
c) Mettre $F000A000_H$ dans le registre R3
d) Diviser par 2 le contenu du registre R4, interprété en signé
e) Multiplier par 17 le contenu du registre R5
f) Multiplier par 7 le contenu du registre R6
g) Multiplier par 119 le contenu du registre R7
h) Mettre à zéro les cases mémoire entre les adresses F0000000 et $F000FFFF_H$

IMPLANTATION MEMOIRE

Soit la déclaration de variables C suivante

```
int a, b;
float x[16], y[16], t[64][64];
short i, j, k;
double z[4] ;
char nom[8];
```

Q 2 : Si l'on suppose que la variable a est à l'adresse 1000 0000H, donnez les adresses hexadécimales des variables x[0], y[0], t[0][0], t[1][0], i , z[0], nom[0] et nom[7]

MICROARCHITECTURES ET TEMPS D'EXECUTION DE PROGRAMMES.

Soient les processeurs non pipelinés (figure 1) et pipelinés (figure 2).

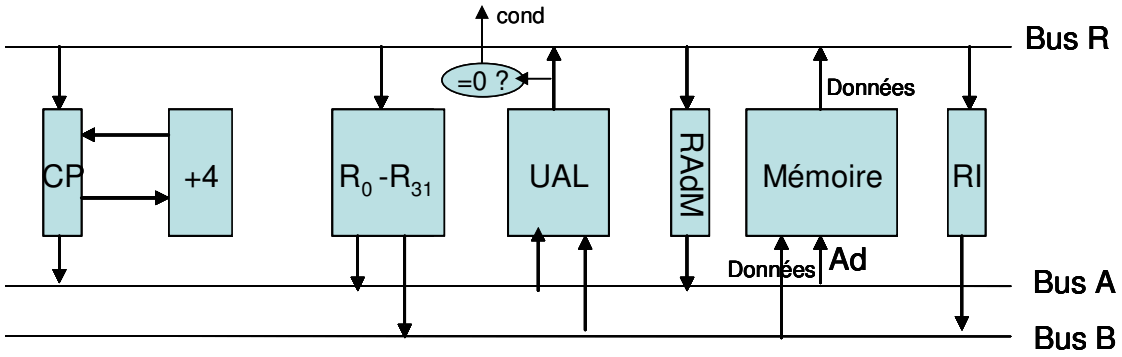

Figure 1 : microarchitecture non pipelinée.

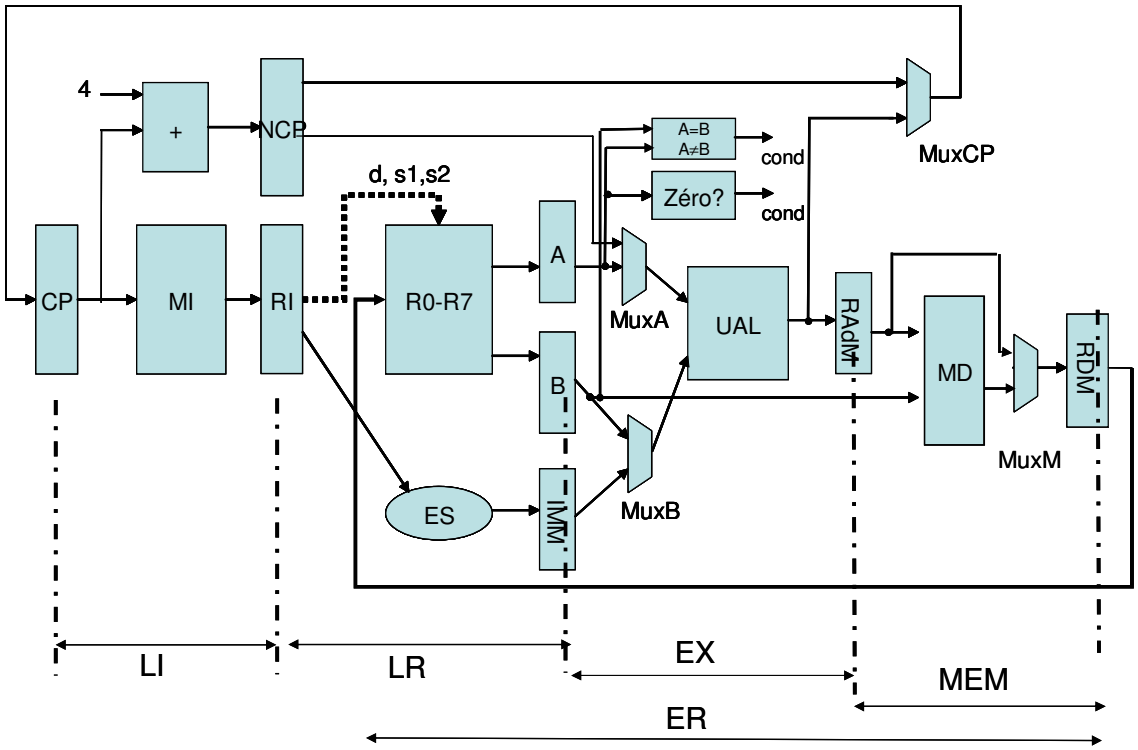

Figure 2 : Microarchitecture pipelinée

Q 3 : Avec le processeur non pipeliné de la Figure 1, donner le temps d'exécution de chacune des instructions suivantes :

(a) ADD Rd,Ra,Rb
(b) ADDI Rd,Ra,Imm
(c) SW Rd, Imm(Ra)
(d) LW Rd, Imm(Ra)
(e) BA Imm où BA Imm signifie : CP <- NCP + ES(Imm**)**
(f) BNEQ Ra, Rb, Imm, dans les cas où Ra=Rb et dans le cas Ra≠Rb

Q 4 : Même question avec le processeur pipeliné de la Figure 2, en supposant qu'aucune anticipation (bypass) n'est implantée. Les instructions après un branchement et qui ont commencé à s'exécuter quand le branchement est pris sont annulées.

Q 5 : Même question avec le processeur pipeliné de la Figure 2, en supposant que toutes les anticipations possibles (bypass) sont implantées. Les instructions après un branchement et qui ont commencé à s'exécuter quand le branchement est pris sont annulées.

Q 6 : Quel est le nombre de cycles par itération pour le programme assembleur MIPS ci-dessous avec :

(a) le processeur non pipeliné de la Figure 1 ;

(b) le processeur pipeliné de la Figure 2, en supposant que toutes les anticipations (bypass) possibles sont implantées et le même comportement des branchements qu'en Q4 et Q5.

Programme assembleur MIPS

```
Boucle :LW R11, (R1)
        LW R12, (R2)
        ADD R11,R11,R12
        SW R11, (R3)
        ADDI R1,R1,4
        ADDI R2,R2,4
        ADDI R3,R3,4
        BNEQ R1,R4, Boucle
```

Optimisation de programmes flottants.

Le processeur utilisé a le jeu d'instructions MIPS (avec branchements non retardés) donné en annexe. Tous les branchements sont parfaitement prédits (s'exécutent en 1 cycle)
La latence des instructions est définie de la manière suivante : une instruction i a une latence de n si l'instruction suivante peut commencer au cycle i+n ; une latence de 1 signifie que l'instruction suivante peut commencer au cycle suivant.
Les instructions flottantes ont les latences suivantes :

Instructions	Latence	Pipelinée ?
LF	2	OUI
FADD, FSUB	3	OUI
FMUL	5	OUI
FDIV	15	NON

Soient les programmes P1, P2 et P3 :

P1	P2	P3
float X[N], Y[N], Z[N] ; int i ; for (i=0 ; i<N ; i++) Z[i]= X[i] + Y[i] ;	float X[N], Y[N], Z[N], a ; int i ; for (i=0 ; i<N ; i++) Z[i]= X[i] /a + Y[i] ;	int i ; float X[N], Y[N], Z[N], a, c ; c =1.0/a ; for (i=0 ; i<N ; i++) Z[i]= X[i] *c + Y[i] ;

Q 7 : Ecrire la version optimisée sans déroulage de boucle du programme P1. Quel est le nombre de cycles par itération et le nombre de cycles total du programme si N=100 ?

Q 8 : Ecrire la version optimisée du programme P1 avec un déroulage de boucle d'ordre 2. Quel est le nombre de cycles par itération et le nombre de cycles total si N=100 ?

Q 9 : Quelle est la version la plus rapide entre le programme P2 et P3 (justifier). Pour la version la plus rapide,

a) écrire la version optimisée sans déroulage de boucle , donner le nombre de cycles par itération et le nombre de cycles total si N=100

b) écrire la version optimisée avec déroulage de boucle d'ordre 2, le nombre de cycles par itération et le nombre de cycles total si N=100.

CACHES.

On suppose que le processeur utilisé a un cache données de 8 Ko, avec des blocs de 64 octets. Il utilise l'écriture simultanée (write through) non allouée. On considère le programmes P1 de l'exercice précédent. On suppose que les tableaux X[N], Y[N], Z[N] sont rangés à partir de l'adresse hexadécimale 1000 0000H.

Q 10: On suppose N= 100. Quelles sont les adresses de X[0], Y[0] et Z[0] ? Dans quels blocs de cache vont X[0], Y[0] et Z[0] ? Quel est le nombre total de défaut de caches lors de l'exécution du programme P1 quand N=100 pour les deux cas suivants : a) correspondance directe, b) associativité deux voies (deux blocs par ensemble) ?

Q 11 : On suppose N= 2048. Dans quels blocs de cache vont X[0], Y[0] et Z[0] ? Quel est le nombre total de défauts de cache pour le programme P1 pour les deux cas : a) correspondance directe, b) associativité 2 voies.

ENTREES-SORTIES

On dispose d'un clavier et d'un écran.
Dans le contrôleur clavier, le mot d'état est à l'adresse ETATIN et le mot de données à l'adresse DATAIN. Le bit « caractère prêt » est b2.
Dans le contrôleur écran, le mot d'état est à l'adresse ETATOUT et le mot de données à l'adresse DATAOUT. Le bit « prêt à émettre » est b1.

Q 12 : Que fait le programme suivant ?

```
L1 :    LB R1, ETATIN
        ANDI R1,R1, 4
        BEQ R1,R0, L1
        LB R1, DATAIN
        SB R1, (R2)
        ADDI R2,R2,1
L2      LB R3, ETATOUT
        ANDI R3,R3,2
        BEQ R3,R0,L2
        SB R1, DATAOUT
        XORI R4, R1, 0xD // caractère ASCII du retour chariot
        BNEQ R4, R0, L1
```

ADD	R	ADD rd, rs, rt	rd ← rs + rt (signé)
ADDI	I	ADDI rt, rs, IMM	rt ← rs + SIMM (signé)
ADDIU	I	ADDIU rt, rs, IMM	rt ← rs + SIMM (le contenu des registres est non signé)
ADDU	R	ADDU rd, rs, rt	rd ← rs + rt (le contenu des registres est non signé)
AND	R	AND rd, rs, rt	rd ← rs and rt
ANDI	I	ANDI rt, rs, IMM	rt ← rs and ZIMM
BEQ	I	BEQ rs,rt, déplac.	si rs = rt, branche à NCP+DÉPLAC. (NCP = adresse instruction suivante)
BGEZ	I	BGEZ rs,déplac.	si rs ≥ 0, branche à NCP+DÉPLAC.
BGEZAL	I	BGEZAL rs, déplac.	NCP dans R31 si rs ≥ 0, branche à NCP+DÉPLAC.
BGTZ	I	BGTZ rs,déplac.	si rs > 0, branche à NCP+DÉPLAC.
BLEZ	I	BLEZ rs,déplac.	si rs ≤ 0, branche à NCP+DÉPLAC.
BLTZ	I	BLTZ rs,déplac.	si rs < 0, branche à NCP+DÉPLAC.
BLTZAL	I	BLTZAL rs, déplac.	NCP dans R31. si rs < 0, branche à NCP+DÉPLAC.
BNEQ	I	BNEQ rs,rt, déplac.	si rs ≠ rt, branche à NCP+DÉPLAC.
J	J	J destination	Décale l'adresse destination de 2 bits à gauche, concatène aux 4 bits de poids fort de CP et saute à l'adresse obtenue
JAL	J	JAL destination	Même action que J . Range NCP dans R31
JALR	R	JALR rs, rd	Saute à l'adresse dans rs. Range NCP dans rd
JR	R	JR rs	Saute à l'adresse dans rs
LUI	I	LUI rt, IMM	Place IMM dans les 16 bits de poids fort de rt. Met 0 dans les 16 bits de poids faible de rt
LW	I	LW rt, déplac.(rs)	rt ← MEM [rs + SIMM]
OR	R	AND rd, rs, rt	rd ← rs or rt
ORI	I	ANDI rt, rs, IMM	rt ← rs or ZIMM
SLL	R	SLL rd, rt, nb	Décale rt à gauche de nb bits et range dans rd
SLT	R	SLT rd, rs, rt	rd ← 1 si rs < rt avec rs signé et 0 autrement
SLTI	I	SLTI rt, rs, IMM	rt ← 1 si rs < SIMM avec rs signé et 0 autrement
SLTIU	I	SLTIU rt, rs, IMM	rt ← 1 si rs < ZIMM avec rs non signé et 0 autrement
SLTU	R	SLTU rt, rs, rt	rd ← 1 si rs < rt avec rs et rt non signés et 0 autrement
SRA	R	SRA rd, rt, nb	Décaler (arithmétique) rt à droite de nb bits et ranger dans rd
SRL	R	SRL rd, rt, nb	Décaler (logique) rt à droite de nb bits et ranger dans rd.
SUB	R	SUB rd, rs, rt	rd ← rs - rt (signé)
SUBU	R	SUBU rd rs, rt	rd ← rs - rt(non signé)
SW	I	SW rt, déplac.(rs)	rt => MEM [rs + IMM]
XOR	R	XOR rd, rs, rt	rd ← rs xor rt
XORI	I	XORI rt, rs, IMM	rt ← rs xor ZIMM

Figure 3 : Instructions entières MIPS utilisées (NB : les branchements ne sont pas retardés)

LF	I	LF ft, déplac(rs)	rt ← MEM [rs +SIMM]
SF	I	SF ft, déplac.(rs)	ft → MEM [rs + SIMM]
FADD	R	FADD fd, fs,ft	fd ← fs + ft (addition flottante simple précision)
FMUL	R	FMUL fd, fs,ft	fd ← fs * ft (multiplication flottante simple précision)
FSUB	R	FSUB fd, fs,ft	fd ← fs - ft (soustraction flottante simple précision)
FDIV	R	FDIV fd,fs,ft	fd ← fs / ft (division flottante simple précision)

Figure 4 : Instructions flottantes ajoutées (Ce ne sont pas les instructions MIPS)