Université Paris Sud
Licence d'informatique

D. Etiemble
2006-07

ARCHITECTURE DES ORDINATEURS
PARTIEL Octobre 2006 (CORRIGÉ)

PARTIE 1 : REPRESENTATION DES NOMBRES

Soit la représentation flottante 13 bits correspondant à la figure 1. L'interprétation est similaire à celle des flottants IEEE simple et double précision. S est le bit de signe. L'exposant est biaisé avec un excès 15. La valeur 0 de la partie exposant est réservée pour la représentation de 0 (partie fractionnaire nulle) et des nombres dénormalisés (partie fractionnaire non nulle). La valeur 31 est réservée pour l'infini (partie fractionnaire nulle) et NaN (partie fractionnaire non nulle). Pour 0<PE<31, un nombre N correspond à $(-1)^S$ x (1,fraction) x $2^{(PE-15)}$ où PE est la partie exposant.

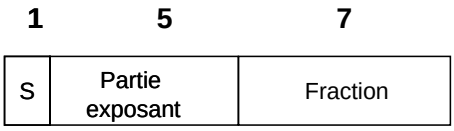


Figure 1 : flottants 13 bits

Pour Q2, Q3 et Q4, les représentations hexadécimales seront données sur 4 digits hexadécimaux.

Q1) Quelle est la plus petite et la plus grande valeur de la mantisse (1,fraction) ?
Plus petite mantisse : 1,0000000 = 1
Plus grande mantisse : 1,1111111 = 1 +1-2^{-7}= 2-2^{-7}= 255/128 = 1,9921875

Q2) Donner les représentations hexadécimales et les valeurs décimales
a) du plus grand nombre positif représentable
011110 1111111 = $0F7F_H$ = (2-2^{-7}) * $2^{(30-15)}$ = 2^{16}-2^8 = 65536 - 256 = 65 280
b) du plus petit nombre positif normalisé représentable (non nul !)
0 00001 000000 = 0080_H = 1 * 2^{-14} = 2^{-14}

Q3) Donner les valeurs décimales ou l'interprétation pour les flottants 13 bits suivants
A) $0A00_H$ +1*$2^{(20-15)}$ = 2^5 = 32
B) $1BC0_H$ - (1,5) * $2^{(23-15)}$ = - 1,5 * 256 = - 384

Q4) Donnez la représentation hexadécimale en flottants 13 bits des nombres
A) +47
47 =(47/32*32) =(1+15/32)*2^5 = (1+1/4+1/8+1/16+1/32)*2^5
= 0 10100 0111100 = $0A3C_H$

B) -65
-65 =-(65/64)*2^6 = -(1+1/64)2^6= 1 10101 0000010 = $1A82_H$

Q5) Donner les résultats en hexadécimal pour les opérations sur les nombres flottants 13 bits :

A) $0A00_H + 0970_H = 32+15 = 47 = 0A3C_H$

B) $0A00 * 0970 = 32 * 15 = 15/8*2^3 * 2^5 = +(1+1/2+1/4+1/8)*2^{(23-15)} = 0BF0_H$

Dans la suite, on utilise le jeu d'instructions décrit en annexe

PARTIE 2 : INSTRUCTIONS

La figure 3 en annexe donne le code opération des instructions du format I et du format J

Q6) Donner sur 8 chiffres hexadécimaux les instructions correspondant aux instructions assembleurs suivantes

a) ADDI R3, R1, -2_{10}

001000 00**001 0**0011 **1111**1111**1111**1110 = $2023FFFE_H$

b) LW R1, (R7+ 128_{10})

100011 00**111 0**0001 **0000**0000**1000**0000 = $8CE10080_H$

Q7) Pour les instructions de type J et I suivantes,

a) entre quelle adresse min et quelle adresse max peut être l'adresse destination de l'instruction JAL si l'adresse de JAL est $FFFF1234_H$?

Adresse min = $F000\ 0000_H$

Adresse max = $FFFF\ FFFC_H$

b) quelle est l'adresse destination de l'instruction BEQ R0,R0, 1000_H si l'adresse de BEQ R0,R0, 1000_H est $F000200C_H$?

NCP = $F000200C + 4 = F0002010_H$

Adresse destination CP = NCP + (1000_H<<2) = $F0002010 + 4000 = F0006010_H$

PARTIE 3 : EXECUTION D'INSTRUCTIONS

On suppose que les registres R1 à R5 ont les contenus suivants, exprimés en hexadécimal :

R1	12345678
R2	FDB97531
R3	0000 000A
R4	8800 0000
R5	A000 0000

Donner les valeurs des registres modifiés après exécution des instructions suivantes

a) AND R6, R2,R1 R6=10305430

b) OR R7, R2,R1 R7 = FFBD7779

c) XOR R8,R2,R1 R8= EF8D2349

Q8) Donner les valeurs des registres modifiés après exécution des instructions suivantes. On indiquera les cas de débordement

a) ADD R9, R1,R2 R9= 0FEDCBA9 OK : P+N=>P
b) ADDU R10,R4,R5 R10 = 2800 0000
Débordement : les nombres sont positifs et la somme est inférieure à chacun des contenus initiaux
c) SRA R12, R5, 4 R12 = FA00 0000 Décalage arithmétiqaue
d) SRL R13,R4, 8 R13 = 0088 0000 Décalage logique

Q9) Soit le programme C suivant

```
int X[100],  S ;

S=0 ;
for (i=0 ; i<100 ; i++)
        if (X[i] >0 && X[i] <21)  s+=X[i] ;
```

On supposera que X[0] est à l'adresse $F0004000_H$

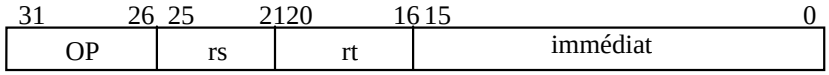
Ecrire le programme en langage assembleur MIPS

	LUI R2, F000	R2 = F0000000
	ORI R2, R2, 4000	R2 = F0004000
	XOR R1,R1,R1	S=0 ;
	ADDI R4,R2,400	Adresse de X[100] hors tableau
Boucle	LW R3, (R2)	Charge X[i]
	BLE R3, Suite	Saut si négatif ou nul
	SLTIU R5, R3, 21	R5 = 1 si R3 <21
	BEQ R5,R0, Suite	
	ADD R1,R1,R3	S+=X[i]
Suite	ADDI R2,R2,4	i=i+1
	BNE R2,R4, Boucle	Si pas fini, branche à Boucle
	SW R1,(R2)	Si S est juste derrière X[99]

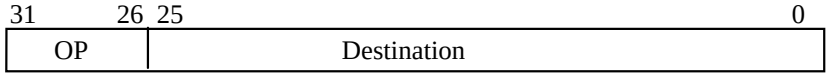
ANNEXE

Le processeur considéré est une version simplifiée du MIPS R2000.
Il a 32 registres entiers de 32 bits, notés R0 à R31. Le registre R0 est câblé à 0.
Il y a trois formats d'instructions (figure 2)

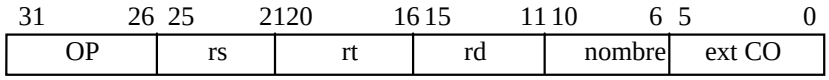
FORMAT I : IMMÉDIAT



FORMAT J : SAUTS



FORMAT R : REGISTRE



OP : code opération
ext CO : extension du code opération

Figure 2 : Formats d'instructions.

Les instructions Load et Store utilisent le mode Immédiat. La mémoire est adressée par octet. Les comparaisons rangent le résultat (booléen vrai ou faux) dans un registre général. Le registre contient 1 (vrai) ou 0 (faux).
Les instructions utilisées sont données dans la figure 3.
IMM est l'immédiat sur 16 bits dans l'instruction.. SIMM est l'immédiat de 16 bits étendu sur 32 bits avec extension de signe.. ZIMM est l'immédiat de 16 bits étendu sur 32 bits avec 16 zéros à gauche. ADBRANCH est l'adresse de l'instruction suivante + SIMM

	Code op (bits 31-26)			
ADDI	001000	I	ADDI rt, rs, IMM	rt <= rs + SIMM (signé)
ADDIU	001001	I	ADDIU rt, rs, IMM	rt <= rs + SIMM (le contenu des registres est non signé)
ANDI	001100	I	ANDI rt, rs, IMM	rt <= rs and ZIMM
BEQ	000100	I	BEQ rs,rt, déplac.	si rs = rt, CP=NCP+(déplac)<<2
BGEZ		I	BGEZ rs,déplac.	si rs ≥ 0, CP=NCP+(déplac)<<2
BGEZAL		I	BGEZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs ≥ 0, CP=NCP+(déplac)<<2
BGTZ		I	BGTZ rs,déplac.	si rs > 0, CP=NCP+(déplac)<<2
BLEZ		I	BLEZ rs,déplac.	si rs ≤ 0, CP=NCP+(déplac)<<2
BLTZ		I	BLTZ rs,déplac.	si rs < 0, CP=NCP+(déplac)<<2
BLTZAL		I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs < 0, CP=NCP+(déplac)<<2
BNEQ	000101	I	BNEQ rs,rt, déplac.	si rs ≠ rt, CP=NCP+(déplac)<<2

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	100011	I	LW rt, déplac.(rs)	rt <= MEM [rs + déplac.]
ORI	001101	I	ANDI rt, rs, IMM	rt <= rs or ZIMM
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
SW	101011	I	SW rt, déplac.(rs)	rt => MEM [rs + déplac.]
XORI	001110	I	XORI rt, rs, IMM	rt <= rs xor ZIMM
J	000010	J	J destination	Décale sur 28 bits l'adresse destination de 2 bits à gauche, concatène aux 4 bits de poids fort de CP et saute à l'adresse obtenue
JAL	000011	J	JAL destination	Même action que J . Range adresse instruction suivante dans R31
ADD		R	ADD rd, rs, rt	rd <= rs + rt (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
JALR		R	JALR rs, rd	Saute à l'adresse dans rs. Range adresse instruction suivante dans rd
JR		R	JR rs	Saute à l'adresse dans rs
OR		R	AND rd, rs, rt	rd <= rs or rt
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
SLTU		R	SLTU rt, rs, r	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)
XOR		R	XOR rd, rs, rt	rd <= rs xor rt

Figure 3 : Jeu d'instructions