

ARCHITECTURE DES ORDINATEURS PARTIEL Octobre 2006

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 $(-1)^S \times (1, \text{fraction}) \times 2^{(PE-15)}$ où PE est la partie exposant.

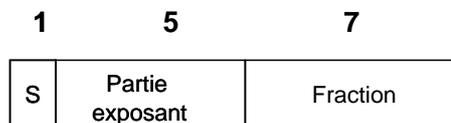


Figure 1 : flottants 13 bits

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) ?

Q2) Donner les représentations hexadécimales et les valeurs décimales

- a) du plus grand nombre positif représentable
- b) du plus petit nombre positif normalisé représentable (non nul !)

Q3) Donner les valeurs décimales ou l'interprétation pour les flottants 13 bits suivants

- A) 0A00_H
- B) 1BC0_H

Q4) Donnez la représentation hexadécimale en flottants 13 bits des nombres

- A) +47
- B) -65

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

- A) 0A00_H + 0970_H =
- B) 0A00_H * 0970_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, -2₁₀
- b) LW R1, (R7+ 128)₁₀

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 ?
- 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?

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

Q8) Donner les valeurs des registres modifiés après exécution des instructions suivantes

- a) AND R6, R2,R1
- b) OR R7, R2,R1
- c) XOR R8,R2,R1

Q9) Donner les valeurs des registres modifiés après exécution des instructions suivantes. On indiquera les cas de débordement

- a) ADD R9, R1,R2
- b) ADDU R10,R4,R5
- c) SRA R12, R5, 4
- d) SRL R13,R4, 8

Q10) 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 F0004000H

Ecrire le programme correspondant en langage assembleur MIPS.

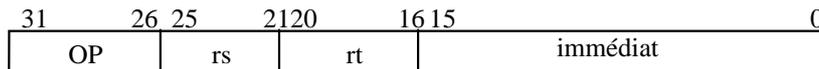
ANNEXE

Le processeur considéré est une version simplifiée du MIPS R2000.

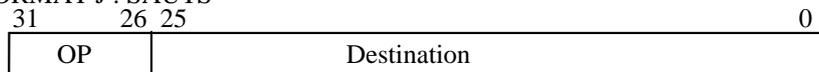
Il a 32 registres entiers de 32 bits, notés R0 à R31. Le registre R0 est câblé à 0.

Il y a trois formats d'instructions (figure 2)

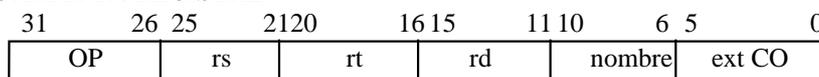
FORMAT I : IMMÉDIAT



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 $rs < 0$, $CP=NCP+(déplac)\ll 2$
BLTZAL		I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31 si $rs < 0$, $CP=NCP+(déplac)\ll 2$
BNEQ	000101	I	BNEQ rs,rt, déplac.	si rs différent rt , $CP=NCP+(déplac)\ll 2$
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)	$rt \leq MEM [rs + déplac.]$
ORI	001101	I	ANDI rt, rs, IMM	$rt \leq rs$ or ZIMM
SLTI		I	SLTI rt, rs, IMM	$rt \leq 1$ si $rs < SIMM$ avec rs signé et 0 autrement
SLTIU		I	SLTIU rt, rs, IMM	$rt \leq 1$ si $rs < ZIMM$ avec rs non signé et 0 autrement
SW	101011	I	SW rt, déplac.(rs)	$rt \Rightarrow MEM [rs + déplac.]$
XORI	001110	I	XORI rt, rs, IMM	$rt \leq rs$ 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	$rd \leq rs + rt$ (signé)
ADDU		R	ADDU rd, rs, rt	$rd \leq rs + rt$ (le contenu des registres est non signé)
AND		R	AND rd, rs, rt	$rd \leq rs$ 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	$rd \leq rs$ 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	$rd \leq 1$ si $rs < rt$ avec rs signé et 0 autrement
SLTU		R	SLTU rt, rs, r	$rd \leq 1$ 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	$rd \leq rs - rt$ (signé)
SUBU		R	SUBU rd rs, rt	$rd \leq rs - rt$ (non signé)
XOR		R	XOR rd, rs, rt	$rd \leq rs$ xor rt

Figure 3 : Jeu d'instructions