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 : Jeu 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

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 ADBRANCH
BGEZ	I	BGEZ rs,déplac.	si rs ≥ 0, branche à ADBRANH
BGEZAL	I	BGEZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs ≥ 0, branche à ADBRANH
BGTZ	I	BGTZ rs,déplac.	si rs > 0, branche à ADBRANH
BLEZ	I	BLEZ rs,déplac.	si rs ≤ 0, branche à ADBRANH
BLTZ	I	BLTZ rs,déplac.	si rs < 0, branche à ADBRANH
BLTZAL	I	BLTZAL rs, déplac.	adresse de l'instruction suivante dans R31 si rs < 0, branche à ADBRANH
BNEQ	I	BNEQ rs,rt, déplac.	si rs ≠ rt, branche à ADBRANCH
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 adresse instruction suivante dans R31
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
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 + IMM]
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, 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.

Q10) Ecrire la suite des instructions qui permet de trouver les valeurs MAX et MIN d'un tableau de 1024 entiers signés rangés à partir de l'adresse $B000\ 0000_H$. On rangera la valeur MAX à l'adresse 0100_H et la valeur MIN à l'adresse 0104_H.

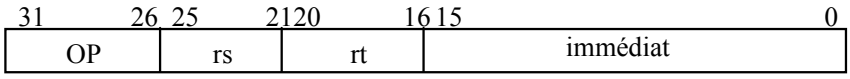
```
        LUI R2, B000H
        ADDI R3, R2, 4*1023₁₀ //adresse de T(1023)
        LW R1,0(R2) // T(0)
        ADD R4, R1, R0 // R4 = min
        ADD R5, R1, R0 // R5 = max
Boucle : ADDI R2,R2,4
        LW R1, 0(R2)
        SLT R6, R4,R1
        BGTZ R6, suite // si Min est inférieur à T(i) aller à suite
        ADD R4, R1, R0 // T(i) est le nouveau min
        BNE R2,R3, Boucle       //on itère
Suite : SLT R6, R1, R5
        BLEZ R6, fin // si T(i) <max, aller à fin
        ADD R5, R1, R0 // T(i) est le nouveau max
Fin     BNEQ R2,R3, Boucle

        SW R5, 0100H(R0)
        SW R4, 0104H(R0)
```

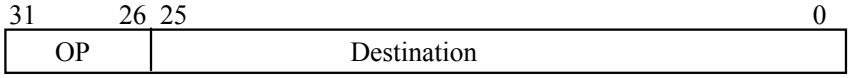
ANNEXE

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)

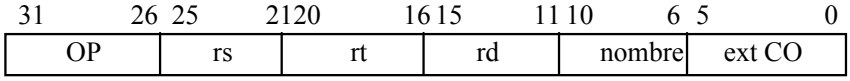
FORMAT I : IMMÉDIAT



FORMAT J : SAUTS



FORMAT R : REGISTRE



OP : code opération
ext CO : extension du code opération

Figure 2 : Formats d'instructions.

Q5) Même question que Q4) avec l'instruction ADDU.

Plus petite valeur : 0000 0000_H soit 0

Plus grande valeur : $FFFFFFFF_H$ soit 2^{32}-1

Q6) On suppose que l'instruction J Adresse_saut est située à l'adresse 8000 8000_H. Indiquer les valeurs min et max que peut prendre Adresse_saut.

Le déplacement est de 26 bits. Il est d'abord multiplié par 4 soit 2^{28}
Il est ensuite concaténé aux 4 bits de poids fort du CP, soit 8
La valeur min de Adresse_saut est 8000 0000_H
La valeur max de Adresse_saut est 8FFF $FFFC_H$

Q7) CP = 8000 0000_H. On veut effectuer un saut à l'adresse DE00 0000_H. Donner la suite des instructions permettant d'effectuer ce saut. Même question si on veut sauter à l'adresse DE00 8800

```
LUI R1,DE00
JR R1
```

```
LUI R1,DE00
ORI R1,R1,8800
JR R1
```

Q8) Ecrire la suite des instructions qui permet de mettre à zéro la zone mémoire comprise entre les adresses A000 0000_H et A000 $FFFF_H$.

```
        LUI R2, A000H
        LUI R3, A001H
Boucle: SW R0, 0(R2)
        ADDI R2,R2,4
        BNE R2,R3, Boucle
```

NB : on utilise des instructions SW (mots). Il faut prendre garde à ce que le test de fin (égalité ou non égalité par rapport à l'adresse du dernier élément du tableau ou de l'élément suivant soit sur une adresse de mot : A000 FFFC si l'on compare au dernier élément du tableau, ou A0010000 si l'on compare à l'adresse du mot qui suit le dernier élément du tableau.

Q9) Reprendre la question Q8) en écrivant une procédure qui met à zéro un tableau de N entiers. Les paramètres de la procédure seront l'adresse du tableau et le nombre d'éléments du tableau.

NB : le respect des conventions logicielles du MIPS pour le passage des paramètres n'est pas exigé.

```
Proc :
        SLLI  R5,R5,2    // 4*N
        ADD R4,R4,R5  // Adresse après le dernier élément du tableau
Boucle :SW R0, 0(R4)
        ADDI R4,R4,4
        BNE R4,R5, Boucle
        JR R31
```

Université Paris Sud D. Etiemble
Licence d'informatique 2004-2005

ARCHITECTURE DES ORDINATEURS
PARTIEL Novembre 2004 (CORRIGÉ)

PARTIE 1 : REPRESENTATION DES NOMBRES

Soit la représentation flottante 16 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 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

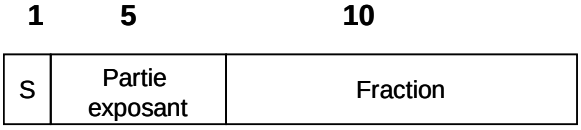


Figure 1 : flottants 16 bits

Q1) Donnez les valeurs décimales pour les flottants 16 bits suivants

A) $5E00_H$ 0101 1110 0000 0000 = $1,5x2^8$= 384

B) 8700_H 1000 0111 0000 0000 = -1,75 x 2^{-14} = -7 x 2^{-16}

Q2) Donnez les valeurs décimales

du plus grand nombre normalisé positif représentable,

0111 1011 1111 1111 = $(2-2^{-10})x\ 2^{15} = 2^{16}-2^5$ = 65 504

du plus petit nombre normalisé positif représentable.

0000 0100 0000 0000 = $1*2^{-14}= 2^{-14}$

Q3) Donnez la représentation hexadécimale en flottants 16 bits des nombres

A) -50

-50=-(50/32) *32 = -(1+1/2+1/16) $*2^5$ = 1 10100 1001000000 = $D240_H$

B) 70 000 > 65504. Il est donc représenté par +∞

0 11111 0000000000 = $7C00_H$

PARTIE 2 : JEU D'INSTRUCTIONS ET PROGRAMMATION ASSEMBLEUR

DANS TOUTE CETTE PARTIE, ON UTILISE LE PROCESSEUR DONT LE JEU D'INSTRUCTIONS ET LE FORMAT D'INSTRUCTIONS SONT DECRITS EN ANNEXE

Q4) On considère l'instruction ADD. Donner la plus grande valeur et la plus petite valeur représentable dans les registres utilisant cette instruction. Pour les deux valeurs, on demande la forme hexadécimale et son équivalent décimal.

Plus petite valeur : $8000\ 0000_H$ soit -2^{31}

Plus grande valeur : $7FFF\ FFFF_H$ soit $2^{31} - 1$