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.
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

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)

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

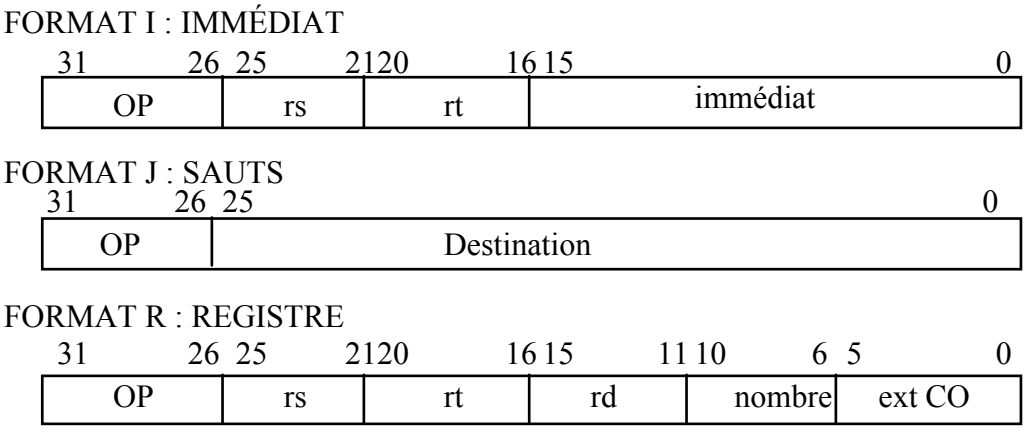

Figure 2 : Formats d'instructions.

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

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.

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.

Utiliser l'algorithme correspondant au programme C suivant :

```
max = tab[0] ;
min= tab[0] ;
for (i=0 ; i < 1024 ; i++) {
        if (tab[i] < min )
           min = tab[i];
        else
            if (tab[i] > max )
                  max = tab[i];
}
```

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

ARCHITECTURE DES ORDINATEURS
PARTIEL Novembre 2004

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 de l'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

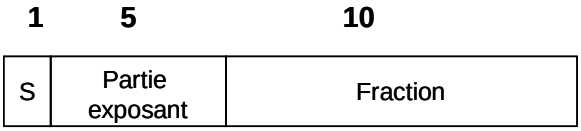

Figure 1 : flottants 16 bits

Q1) Donnez les valeurs décimales pour les flottants 16 bits suivants
- a) $5E00_H$
- b) 8700_H

Q2) Donnez les valeurs décimales
du plus grand nombre normalisé positif représentable,
du plus petit nombre normalisé positif représentable.

Q3) Donnez la représentation hexadécimale en flottants 16 bits des nombres
- A) -50
- B) +70000

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 lorsque cette instruction est utilisée. Pour les deux valeurs, on demande la forme hexadécimale et son équivalent décimal.

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

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.

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