

ARCHITECTURE DES ORDINATEURS  
Corrigé PARTIEL Octobre 2009  
Tous documents autorisés - 2H

**Pour toutes les questions, on utilise le jeu d'instructions NIOS-II. Les différentes parties sont indépendantes**

**PARTIE 1 : Implantation mémoire**

Soit la déclaration de variables suivante pour un programme C  
char Toto[11] ;  
int X,Y,Z ;  
short A,B,C ;  
double D[10],E;  
float F,G,H

Q 1) En supposant que toto[0] soit à l'adresse F000 0000<sub>H</sub>, donnez les adresses (hexadécimale) de X, A, D[0], et H. Quelle est l'occupation mémoire en octets ?

Variables	Adresse relative	Adresse
toto [0]	0	F000 0000
toto [10]	10	F000 000A
X, Y, Z	12, 16, 20	<b>F000 000C</b> , F000 0010, F000 0014
A,B,C	24,26,28	<b>F000 0018</b> , F000 001A, F000 001C
D[0]	32	<b>F000 0020</b>
D[9]	104	F000 0068
E	112	F000 0070
F,G,H	120,124,128	F000 0078, F000 007C, <b>F000 0080</b>
Total	132 octets	

**Q 2) Proposer une déclaration qui minimise l'occupation mémoire et donnez le nombre d'octets correspondant à cette nouvelle déclaration..**

Int X,Y,Z ;  
Double D[10],E;  
Float F,G,H  
Short A,B,C ;  
Char toto[11] ;

Int = 12 octets ; Double = 88 octets ; Float = 12 octets ; Shorts = 6 octets ; Char = 11 octets  
Sans problèmes d'alignement  
Total = 129 octets.

**PARTIE 2 : Exécution d'instructions**

Les registres du processeur NIOS contiennent les valeurs suivantes en hexadécimal.

R0	0000 0000
R1	89AB 1234
R2	5432 9876
R3	FFFF FFFF
R4	ABEF CD01

**Q 3) Donner le contenu des registres R5 à R10 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes.**

- |                   |                |
|-------------------|----------------|
| a) ADD R5, R2, R1 | R5= DDDD AAAA  |
| b) SUB R6, R1, R2 | R6 = 3578 79BE |
| c) SLLI R7,R1, 4  | R7 = 9AB1 2340 |
| d) SRAI R8, R1,8  | R8 = FF89 AB12 |
| e) SRLI R9, R1, 1 | R9 = 88D5 891A |
| f) MUL R10, R1,R3 | R10= 7654EDCC  |

**Q 4) Donner le contenu des registres R11 à R15 (sous forme de huit chiffres hexadécimaux) après exécution des instructions suivantes.**

- |                     |                              |
|---------------------|------------------------------|
| a) OR R11, R1, R2   | R11 = DDBB 9A76              |
| b) AND R12, R1, R2  | R12 = 0022 1034              |
| c) XOR R13, R1, R2  | R13 = DD99 8 <sup>a</sup> 42 |
| d) CMPGE R14, R0,R1 | R14 = 0000 0001              |
| e) CMPLT R15,R3,R4  | R15 = 0000 0000              |

**Q 5) Donner le contenu (sous forme de huit chiffres hexadécimaux) du registre CP après exécution des instructions suivantes en supposant à chaque fois que l'adresse de l'instruction est : 1000 0000<sub>H</sub>.**

- |                  |                 |
|------------------|-----------------|
| a) BEQ R0,R0, +8 | CP = 1000 000C  |
| b) BNE R0,R0, +4 | CP = 1000 0004. |

### PARTIE 3 : Désassemblage

Soit le programme assembleur P1 qui travaille sur un tableau d'entiers T[N] dont l'adresse de début est contenue dans le registre R1 et range le résultat dans la case mémoire d'adresse 00007000<sub>H</sub>

	ADDI R2,R2, N // N <2 <sup>15</sup>
	SLLI R2,R2,2
	ADD R2,R1,R2
	ADD R3,R0,R0
Boucle	LDW R4, 0(R1)
	ADDI R1,R1,4
	BNE R4,R0, Suite
	ADDI R3,R3,1
Suite	BNE R1;R2, Boucle
	STW R3, 7000 <sub>H</sub> (R0)

**Q 6) Que fait le programme P1 ?**

Le programme compte le nombre d'éléments nuls du tableau d'entiers T[N ] et place le résultat dans la case mémoire d'adresse 7000H..

**Q 7) Donner le programme C correspondant au programme P1**

```
Int T[N], i, RES =0 ;
For (i=0 ;i<N ; i++)
    If (T[i] ==0) RES++ ;
```

**PARTIE 4 : Programmation assembleur**

**Q 8) Ecrire un programme en assembleur NIOS qui a dans R4 (partie basse) et R5 (partie haute) un entier non signé sur 64 bits d'une part, dans R6 (partie basse) et R7 (partie haute) un second entier non signé de 64 bits et renvoie le plus grand de ces deux entiers non signés dans R2 (partie basse) et R3 (partie haute).**

Il faut comparer les parties hautes (comparaison non signée)  
 Si R5 > R7 alors (R5-R4) > (R7-R6).  
 Si R7 >R5, alors (R7-R6 > (R5-R4)  
 Si R5=R7, alors il faut comparer les parties basses.

	ADD R2,R4,R0	R5,R4 est le max
	ADD R3,R5,R0	
	BGTU R5,R7, FIN	Si R5>R7, c'est vrai
	BNEQ R5,R7, Suite	Si R5≠R7 (R5<R7) alors R6,R7 max : permuter
	BGTU R4,R6, FIN	R5=R7 et R4>R6 en non signé, alors R5,R4 max (vrai)
	ADD R2,R6, R0	
	ADD R3,R7,R0	R6,R7 est le max.
FIN		

**Q 9) Indiquer les instructions qu'il faut ajouter au programme de la question Q8 pour le transformer en une procédure MAXU64**

MAXU64	ADD R2,R4,R0	R5,R4 est le max
	ADD R3,R5,R0	
	BGTU R5,R7, FIN	Si R5>R7, c'est vrai
	BNEQ R5,R7, Suite	Si R5≠R7 (R5<R7) alors R6,R7 max : permuter
	BGTU R4,R6, FIN	R5=R7 et R4>R6 en non signé, alors R5,R4 max (vrai)
	ADD R2,R6, R0	
	ADD R3,R7,R0	R6,R7 est le max.
FIN	JMP R31	

**Q 10) Ecrire une procédure NIOSII MAXS64 qui reçoit dans R4 (partie basse) et R5 (partie haute) un entier signé sur 64 bits d'une part, dans R6 (partie basse) et R7 (partie haute) un second entier signé de 64 bits et renvoie le plus grand de ces deux entiers signés dans R2 (partie basse) et R3 (partie haute)**

Il faut comparer les parties hautes (comparaison signée)

Si  $R5 > R7$  alors  $(R5-R4) > (R7-R6)$ .

Si  $R7 > R5$ , alors  $(R7-R6) > (R5-R4)$

Si  $R5=R7$ , alors il faut comparer les parties basses, mais avec une comparaison **non** signée.

Ceci résulte de la représentation des nombres en complément à 2

$$N = -b_{63} 2^{63} + \sum_0^{62} b_i 2^i$$

Si les bits de poids fort (63 à 32) sont identiques, ce sont les bits 31 à 0 (en non signé) qui vont déterminer le plus grand.

MAXS64	ADD R2,R4,R0	R5,R4 est le max
	ADD R3,R5,R0	
	BGT R5,R7, FIN	Si $R5 > R7$ en signé, c'est vrai
	BNEQ R5,R7, Suite	Si $R5 \neq R7$ ( $R5 < R7$ ) alors R6,R7 max : permuter
	<b>BGTU</b> R4,R6, FIN	$R5=R7$ et $R4 > R6$ en non signé, alors R5,R4 max (vrai)
	ADD R2,R6, R0	
	ADD R3,R7,R0	R6,R7 est le max.
FIN	JMP R31	