

TD n° 1 : MODELES D'EXECUTION INSTRUCTIONS ARITHMETIQUES ET LOGIQUES

1. Quatre modèles d'exécution

Le modèle d'exécution des instructions est donné par le couple (n, m) où n est le nombre d'opérandes spécifié par instruction, et m est le nombre d'opérandes mémoire.

Soient les quatre modèles

- a) **Modèle (3,0) : machine chargement –rangement.** Les accès mémoire ne sont possibles que par les instructions de chargement (Load) et rangement (Store). Les instructions arithmétiques et logiques ne portent que sur des opérandes situés dans des registres.
- b) **Modèle (1,1) : machine à accumulateur.** Un seul opérande de type mémoire est spécifié dans les instructions arithmétiques et logiques, l'autre étant un registre (implicite) : l'accumulateur
- c) **Modèle (2,1) : les instructions arithmétiques et logiques ont deux opérandes,** l'un dans un registre (qui est à la fois source et destination) et l'autre étant un opérande mémoire. Le modèle (2,1) inclut le modèle (2,0) où les deux instructions sont dans des registres
- d) **Modèle (0,0) : machine à pile.** Dans une machine à pile, une instruction arithmétique ou logique dépile les deux opérandes en sommet d'une pile, effectue l'opération et empile le résultat. L'instruction Push empile un opérande mémoire. L'instruction Pop dépile un opérande mémoire.

Les instructions disponibles dans les quatre processeurs considérés avec ces modèles d'exécution sont données dans la table ci-dessous. Pour les instructions mémoire, on ne se préoccupe pas des modes d'adressage. Pour les multiplications, on considère que le produit de deux registres ou d'un registre et d'un mot mémoire peut être contenu dans le registre résultat. :

M0 (0,0)	M1 (1,1)	M2 (2,1)	M3 (3,0)
PUSH X	LOAD X (accu ← X)	LOAD Ri,X (Ri ← X)	LOAD Ri,X
POP X	STORE X (X ← accu)	STORE Ri,X (X ← Ri)	STORE Ri,X
ADD	ADD X (accu ← accu + X)	ADD Ri,X (Ri ← Ri + X)	ADD Ri,Rj,Rk (Ri ← Rj+Rk)
SUB	SUB X (accu ← accu - X)	SUB Ri,X (Ri ← Ri - X)	SUB Ri,Rj,Rk (Ri ← Rj - Rk)
MUL	MUL X (accu ← accu * X)	MUL Ri,X (Ri ← Ri * X)	MUL Ri,Rj,Rk (Ri ← Rj * Rk)

Les variables A, B, C, D sont initialement en mémoire.

Q 1) Ecrire les séquences de code pour les quatre machines pour $A = B + C$. Donner le nombre d'instructions et le nombre d'accès mémoire

Q2) Ecrire les séquences de code pour les quatre machines pour la suite d'instructions suivantes. Donner le nombre d'instructions et le nombre d'accès mémoire

$$\begin{aligned}A &= B + C ; \\B &= A + C ; \\D &= A - B ;\end{aligned}$$

Q3) Ecrire les séquences de code pour les quatre machines pour l'expression. Donner le nombre d'instructions et le nombre d'accès mémoire

$$W=(A+B)(C+D) + (D.E)$$

2. Instructions arithmétiques et logiques

Pour cette partie, on utilise le jeu d'instructions NIOS II.

Q 4) On considère l'instruction ADD. Donner la plus grande valeur et la plus petite valeur représentable dans les registres en supposant que les registres contiennent des entiers signés en complément à deux. Pour les deux valeurs, on demande la forme hexadécimale et son équivalent décimal.

Q 5) Même question que Q 4) en supposant que les registres contiennent des entiers non signés.

On suppose maintenant que les registres du processeur contiennent initialement les valeurs données dans la table 1 :

Registre	Contenu (hexa)
R0	00000000
R1	30001000
R2	50001016
R3	8432A380
R4	ECDF1234
R5	89765432
R6	C2345678
R7	A0123456

Table 1

Q 6) En partant à chaque fois du contenu de la table 1, donner le contenu des registres après exécution des instructions NIOS II. On considérera que les registres contiennent des entiers signés en complément à deux. Dans chaque cas, on considérera que l'instruction est à l'adresse 1000 0000_H

- a) ADD R8,R1,R2
- b) ADD R9,R6,R7
- c) SUB R10, R1,R2
- d) SUB R11, R6,R7

Indiquer si le résultat est correct ou non ;

En supposant que les registres contiennent maintenant des entiers non signés, donner le contenu des registres après exécution des instructions

- e) ADD R8,R1,R2
- f) ADD R9,R6,R7

Indiquer si le résultat est correct ou non

Q 7) En prenant en compte les bits de signe des deux opérandes source et le bit de signe du résultat, définir à quelle condition le résultat de l'addition de deux nombres signés est correct ?

Soit le squelette suivant d'un programme assembleur NIOS II. Ecrire la suite d'instructions telle que le programme se termine à l'étiquette STOP si le résultat est correct et à l'étiquette DEB si le résultat est incorrect (débordement). Les registres r1 et r2 contiennent des entiers signés en complément à deux.

```

    add r6,r1,r2
//partie à compléter
STOP: br STOP
DEB : br DEB

```

Q 8) A quelle condition le résultat de l'addition de deux nombres non signés est correct ?

Soit le squelette suivant d'un programme assembleur NIOS II. Ecrire la suite d'instructions telle que le programme se termine à l'étiquette STOP si le résultat est correct et à l'étiquette DEB si le résultat est incorrect (débordement). Les registres r3 et r4 contiennent des entiers non signés.

```

    add r7,r3,r4
//partie à compléter
STOP: br STOP
DEB : br DEB

```

Q 9) Sans utiliser l'instruction de multiplication, écrire la suite d'instructions permettant de ranger dans R2 les valeurs suivantes

- a) R1*8
- b) R1*10
- c) R1*31

3. Variante ARM (optionnel)

Le processeur ARM dispose de 16 registres de 32 bits (dont R15≡PC et R0 n'est pas câblé à 0).

Les instructions arithmétiques et logiques sont du type

$Rd \leftarrow Rs \text{ opération Opérande 2}$.

L'une des manière d'obtenir l'opérande 2 est Opérande 2 = Rm décalé de imm5 position, où Rm est un deuxième registre source et imm5 une constante sur 5 bits.

Par exemple, ADD R0, R1, R2 LSL #4 range dans R0 le résultat de $R1 + (R2 \ll 4)$

Extrait des opérations arithmétiques disponibles

Instructions arithmétiques	ADD, SUB	$Rd \leftarrow Rs \text{ op opérande 2}$
Instructions arithmétiques	RSB	$Rd \leftarrow \text{opérande 2} - Rs$
Instructions logiques	AND, ORR, EOR	$Rd \leftarrow Rs \text{ op opérande 2}$
Instructions de transfert	MOV, MVN	$Rd \leftarrow \text{opérande 2}$ $Rd \leftarrow \text{complément de opérande 2}$

Q 10) Sans utiliser l'instruction de multiplication, écrire la suite d'instructions ARM permettant de ranger dans R2 les valeurs suivantes

- a) R1*8
- b) R1*10
- c) R1*31