

## TD n° 3 : 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 de 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 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.

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

b) Ecrire les séquences de code pour les quatre machines pour  
 $A = B + C$  ;  
 $B = A + C$  ;  
 $D = A - B$  ;

Donner le nombre d'instructions et le nombre d'accès mémoire

c) Ecrire les séquences de code pour les quatre machines pour l'expression  
 $W = (A+B)(C+D) + (D.E)$

Donner le nombre d'instructions et le nombre d'accès mémoire

## 2. Instructions arithmétiques et logiques

### Instructions arithmétiques avec des immédiats

Avec le format d'instruction du MIPS, quelle est l'amplitude des constantes que l'on peut avoir dans les instructions arithmétiques avec immédiats ? Même question avec le jeu d'instructions ARM.

### Pseudo-instructions

En assembleur, une pseudo-instruction correspond à une instruction qui n'est pas présente dans le jeu d'instructions du processeur, mais qui est synthétisée à l'aide d'une ou plusieurs instructions machine. Elle a pour but de simplifier et rendre plus lisible l'écriture de programmes en assembleur

Avec le jeu d'instructions MIPS, donner l'instruction ou la séquence d'instructions processeur pour réaliser les pseudo-instructions suivantes :

- NOP
- CLEAR Ri                    // Ri ← 0
- MOV Ri, Rj                 // Ri ← Rj
- NEG Ri, Rj                 // Ri ← -Rj
- NOT Ri, Rj                 // Ri ← ~Rj (complément bit à bit)

Avec le jeu d'instructions ARM, donner l'instruction ou la séquence d'instructions processeur pour les pseudo-instructions suivantes :

- NOP
- ADR Ri, adresse // adresse située à moins de 256 octets de la valeur courante de CP.

### Permutation de deux registres

Quelle opération logique permet en trois instructions d'échanger le contenu de deux registres R1 et R2 sans utiliser un troisième registre ou un emplacement mémoire

### Multiplication par des constantes

L'instruction de multiplication sur des entiers prend plus d'une dizaine de cycles d'horloge sur la plupart des processeurs. Les instructions arithmétiques et logiques prennent généralement un seul cycle d'horloge. Il est donc plus efficace d'implanter la multiplication du contenu d'un registre par une constante en utilisant des opérations comme l'addition, la soustraction et les décalages.

En utilisant les instructions fournies en annexe pour le MIPS et pour l'ARM, donner

la suite d'instructions MIPS

la suite d'instructions ARM

pour effectuer la multiplication du contenu du registre R1 par

la constante 33

la constante 37

la constante 105

## 3. Utilisation des modes d'adressage

On rappelle que le MIPS dispose du mode d'adressage Adresse = (Ri + déplacement) et de l'instruction load

LD Rd, (Rs+dep)                    Rd ← Mem32 (Rs + ES, dep16)

Les instructions de rangement sont symétriques des instructions de chargement. Les instructions d'addition et soustraction sont disponibles en mode reg+reg et reg + imm16.

L'ARM dispose de plusieurs modes d'adressage pour l'instruction LDR avec la syntaxe assembleur donnée dans la table ci-dessous. Les instructions de rangement sont symétriques des instructions de chargement. Les instructions d'addition et soustraction sont disponibles en mode reg+reg et reg + imm8.

Mode	Assembleur	Action
Déplacement 12 bits, Pré-indexé	[Rn, #déplacement]	Adresse = Rn + déplacement
Déplacement 12 bits, Pré-indexé avec mise à jour	[Rn, #déplacement] !	Adresse = Rn + déplacement Rn ← Adresse
Déplacement 12 bits, Post-indexé	[Rn], #déplacement	Adresse = Rn Rn ← Rn + déplacement
Déplacement dans Rm Préindexé	[Rn, ± Rm, décalage]	
Déplacement dans Rm Préindexé avec mise à jour	[Rn, ± Rm, décalage] !	
Déplacement dans Rm Postindexé	[Rn], ± Rm, décalage	
Relatif		Adresse = CP + déplacement

Soient les deux boucles :

B1  
For (i=0; i<1000; i++)  
    s = s + X[i] + Y[i]

B2  
X[0]=0; X[1]=1;  
For (i=1; i<999; i++)  
    X[i+1] = X[i] + X[i-1]

On suppose que les vecteurs X et Y sont des entiers (32 bits), implantés à partir des adresses 1000 0000H et 2000 0000H. La variable s est placée à l'adresse 100H.

Ecrire le corps de chacune des boucles en assembleur pour le MIPS, puis en utilisant les modes d'adressage de l'ARM.

On supposera que le registre R1 contient l'adresse 10000000H et R2 contient l'adresse 20000000H.

## 4. ANNEXE

### Instructions MIPS (extrait)

31-26	25-21	20-16	15-0
<i>Code op</i>	<i>s</i>	<i>t</i>	<i>déplacement</i>

Format des instructions RI (registre, immédiat)

Instructions arithmétiques et logiques	ADDI, ADDIU, ORI, ANDI, XORI	Rt ← Rs op Immédiat
Instructions de décalage	SLL, SRA, SRL	Rt ← Rs décalage Immédiat

31-26	25-21	20-16	15-11	10-6	5-0
<i>opcode (000000)</i>	<i>s</i>	<i>t</i>	<i>d</i>	<i>Nb de décalages</i>	<i>Extension Code op</i>

Format des instructions RR (registre, registre)

Instructions arithmétiques et logiques	ADD, ADDU, SUB, OR, NOR, AND, XOR	Rd ← Rs op Rt
Instructions de décalage	SLLV, SRAV, SRLV	Rd ← Rs décalage Rt (5bits)

**Instructions ARM (extrait)**

Format des instructions arithmétiques et logiques

31-28			24-21	20	19-16	15-12	11-0
Cond	00	I	Code Op	S	n	d	Opérande 2

Rs est le registre source (s est le numéro)

Rd est le registre destination (d est le numéro)

Lorsque I =0, opérande 2 est obtenu comme suit

Opérande 2 = rotation à droite de 2 n positions [immédiat non signé sur 8 bits]

Où les 12 bits sont interprétés comme suit

11-8	7-0
Rotation (2 x n)	Immédiat

Lorsque I=1, opérande 2 est obtenu comme suit

Opérande 2 = décalage [Rm]

Où les 12 bits sont interprétés comme suit

11-4	3-0
Décalage (nb de bits)	m (numéro de registre)

Les instructions ARM sont définies à la fois pour les formats RR et RI : Opérande 2 = immédiat ou décalage [Rm]

Instructions arithmétiques	ADD, SUB	$Rd \leftarrow Rs \text{ op } \text{opérande 2}$
Instructions arithmétiques	ADC, SUBC	$Rd \leftarrow Rs + \text{opérande 2} + \text{retenue}$ $Rd \leftarrow Rs - \text{opérande 2} + \text{retenue} - 1$
Instructions arithmétiques	RSB, RSC	$Rd \leftarrow \text{opérande 2} - Rs$ $Rd \leftarrow \text{opérande 2} - Rs + \text{retenue} - 1$
Instructions logiques	AND, ORR, EOR	$Rd \leftarrow Rs \text{ op } \text{opérande 2}$
Instructions de transfert	MOV, MVN	$Rd \leftarrow \text{opérande 2}$ $Rd \leftarrow \text{complément de opérande 2}$