TD n° 1 : JEUX D'INSTRUCTIONS
Modèles d'exécution
Taille du code sur des exemples RISC (MIPS et ARM)

1. Introduction

Ce TD n'a pas pour objectif de programmer en assembleur MIPS et ARM. Son objectif est, sur de petits exemples,

1. de comparer le nombre d'instructions et le nombre d'accès mémoire pour différents modèles de jeux d'instructions, dont notamment le modèle (2,1) utilisé dans IA-32 et le modèle (3,0) des jeux d'instructions RISC
2. de comparer le nombre d'instructions pour deux jeux d'instructions RISC (MIPS et ARM) en fonction de certaines caractéristiques des instructions arithmétiques, des instructions de branchements conditionnels et des modes d'adressage mémoire.

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

a) Ecrire les séquences de code pour les quatre machines pour la suite d'instructions

A = B + C ;
B = A + C ;
D = A -B ;

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

b) 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

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 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)

3. Exemples MIPS et ARM pour les instructions arithmétiques

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 ci-dessous 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 31
la constante 37

Instructions MIPS (extrait)

Format des instructions RI

31-26	25-21	20-16	15-0
Code op	*s*	*t*	*Imm16*

Instructions arithmétiques	ADDI	Rt ← Rs + SIMM16
Instructions logiques	ORI, ANDI, XORI	Rt ← Rs op ZIMM16
Instructions de décalage	SLL, SRA, SRL	Rt ← Rs décalage IMM5

Format des instructions RR

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

Instructions arithmétiques et logiques	ADD, 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)
Le format général d'une instruction est

Rd ← Rs opération Opérande 2.

On ne traite pas ici le cas I=0
Lorsque I=1, Opérande 2 = décalage [Rm]

Exemple ADD R2, R1, R0 LSL #4 signifie R2 ← R1 + (R0 <<4) = R1 + 16*R0

LSL est le mnémonique pour le décalage logique à gauche

Instructions arithmétiques	ADD, SUB	Rd ← Rs +/- opérande 2
Instructions arithmétiques	RSB	Rd ← opérande 2 - Rs
Instructions logiques	AND, ORR, EOR	Rd ← Rs op opérande 2
Instructions de transfert	MOV	Rd ← opérande 2

4. Exemples MIPS et ARM sur les branchements conditionnels

Ecrire la séquence d'instruction qui place dans un registre la valeur absolue du contenu de ce registre

- pour le MIPS
- pour ARM sans utiliser les instructions conditionnelles, puis en utilisant les instructions conditionnelles.

Instructions pour les branchements conditionnels (extraits)

MIPS

SLT, SLTU	SLT Rd, Rs1, Rs2 (signé)	Rd ←1 si Rs1<Rs2 et Rd ← 0 sinon
BEQ / BNE	BEQ Rs1, Rs2, imm16	Si Rs1 cond Rs2 alors PC ← NPC + simm16
BLEZ, BGTZ, BLTZ, BGEZ	Bcond Rs1, imm16	Si Rs1 cond 0 alors PC ← NPC + simm16

ARM

CMP , TST	CMP Rs1, Rs2	Rs1-Rs2 → Rcc
Bcond (LT, LE, GT, GE, EQ, NE…)	Bcond, déplacement	Si cond, alors PC ← NPC +simm16

5. Utilisation des modes d'adressage

Le MIPS dispose du mode d'adressage Adresse = Ri + SIMM16 où IMM16 est une constante sur 16 bits (format RI) et SIMM16 est la constante 32 bits obtenue par extension de signe pour les instructions Load et Store

LW Rd, (Rs+imm16) Rd ← Mem32 (Rs + SIMM16)
SW Rd, (Rs+imm16) Rd → Mem32 (Rs + SIMM16)

L'ARM dispose de plusieurs modes d'adressage pour l'instruction Load (LDR) avec la syntaxe assembleur donnée dans la table ci-dessous. Les instructions de rangement sont symétriques des instructions de chargement.

Mode	Assembleur	Action
Déplacement 12 bits, Pré-indexé	[Rn, #deplacement]	Adresse = Rn + déplacement
Déplacement 12 bits, Pré-indexé avec mise à jour	[Rn, #deplacement] !	Adresse = Rn + déplacement Rn ← Adresse
Déplacement 12 bits, Post-indexé	[Rn], #deplacement	Adresse = Rn Rn ← Rn + déplacement

Soit la boucle :

```
for (i=0; i<1000; i++)
    s = s + X[i] + Y[i]
```

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

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 10000000_H et R2 contient l'adresse 20000000_H.