

ARCHITECTURE DES ORDINATEURS
Examen Décembre 2011
3H - Tous documents autorisés
Les questions sont indépendantes

On utilise le jeu d'instructions ARM.

PROGRAMMATION ASSEMBLEUR

PREMIERE PARTIE

Soit le code C

```
int a, b;
short c;
unsigned char d; // Les adresses de a, b, c, d sont initialement
                // dans R1, R2, R3, R4

a= (int) c;
b= (int) d
```

Les instructions nécessaires et les modes d'adressage du jeu d'instructions ARM sont rappelés en annexe.

Question 1) Ecrire le code assembleur ARM correspondant au code C

DEUXIEME PARTIE

Soit le code assembleur ARM

```
        MOV R3, 6
        MOV R1, #0
        STR R1, [R4], 4
        MOV R2, #1
        STR R2, [R4], 4
        SUBS R3, R3, #2
Boucle: ADD R5, R1, R2
        MOV R1, R2
        MOV R2, R5
        STR R5, [R4], 4
        SUBS R3, R3, #1
        BGT Boucle
```

Question 2) En supposant que R4 contient au départ l'adresse d'un tableau T[6], donner le contenu du tableau en fin d'exécution.

TROISIEME PARTIE

On suppose que les registres ARM R0, R1 et R2 contiennent respectivement les variables (int) x, y et s.

Soit les programmes assembleurs P1 et P2

P1)		P2)
MOV R2, #0		MOV R2, #0
CMP R0, #4		CMP R0, #7
CMPEQ R1, #5		CMPNE R1, #5
MOVEQ R2, #3		MOVEQ R2, #5

Question 3) Donner le code C correspondant aux programmes P1 et P2

CACHES

On suppose que le processeur utilisé a un cache données de 64 Ko, avec des lignes de 64 octets.

Le cache utilise la réécriture avec écriture allouée (**il y a des défauts de cache en écriture**)

Le processeur a des adresses sur 32 bits.

Le programme travaille sur deux tableaux de doubles X[N] et Y[N]

&X[0] est F000 0000_H

&Y[0] = &X[N-1] + 8

Question 4) Quel est pour ce cache le nombre de bits pour l'adresse dans la ligne, le nombre de bits d'index et le nombre de bits d'étiquette dans les deux cas suivants : a) correspondance directe, b) associativité quatre voies (quatre lignes par ensemble) avec remplacement par LRU.

Question 5) Dans quelles lignes vont les flottants X[0] et Y[0] en correspondance directe pour N=128, N= 1024 et N=8192 ?

Question 6) Quel est le nombre total de défauts de caches lors de l'exécution des boucles P1, P2 et P3 pour les deux cas suivants : a) correspondance directe, b) associativité quatre voies

Programme P1

```
double X[8192], Y[8192] ;
for (i=0 ; i<8192 ; i++)
    Y[i] = X[i];
```

Programme P2

```
double X[1024], Y[1024] ;
for (i=0 ; i<1024 ; i++)
    Y[i] = X[i] + Y[i] ;
```

Programme P3

```
double X[1024], Y[1024] ;
for (i=0 ; i<1020 ; i++)
    Y[i] =X[i] + X[i+4];
```

EXECUTION DE BOUCLES

On ajoute au jeu d'instructions ARM des instructions flottantes simple précision (32 bits) (Figure 1) et 32 registres flottants F0 à F31 (F0 est un registre normal). Les additions, soustractions et multiplications flottantes sont pipelinées. Une nouvelle instruction peut

démarrer à chaque cycle. Les latences sont de 2 cycles pour LF et de 3 cycles pour les instructions flottantes. Les branchements ne sont pas retardés.

LF	2	LF ft, adresse	ft ← MEM [adresse]
SF	1	SF ft, adresse	ft → MEM [adresse]
FADD	3	FADD fd, fs,ft	fd ← fs + ft (addition flottante simple précision)
FSUB	3	FSUB fd, fs,ft	fd ← fs - ft (soustraction flottante simple précision)
FMUL	3	FMUL fd, fs,ft	fd ← fs * ft (multiplication flottante simple précision)

Figure 1: Instructions flottantes ajoutées (Ce ne sont pas les instructions ARM). Les instructions LF et SF ont les mêmes modes d'adressage que les instructions entières

Question 7) Sans optimisation, quel est le temps d'exécution (en cycles par itération) et le temps d'exécution total du programme de la table 1. (Au démarrage R3 contient l'adresse du tableau X et R4 contient l'adresse du tableau Y)

Table 1 : Programme C et programme assembleur

float X[100], Y[100], S; int i ; for (i=0; i<100; i++) S+=X[i]*X[i] + Y[i]*Y[i];	MOV R5, #100 FSUB F0, F0, F0 Boucle :LF F1, [R3], 4 LF F2, [R4], 4 FMUL F1,F1,F1 FMUL F2,F2,F2 FADD F0,F0,F1 FADD F0,F0,F2 SUBS R5,R5,#1 BGT Boucle SF F0, (adresse de S)
---	---

Question 8) Optimiser la boucle de la table 1 et donner le nouveau temps d'exécution.

PIPELINE

Un processeur a les pipelines suivants :

Instructions arithmétiques	LI1	LI2	DI	LR	EX1	EX2	ER	
Lecture mémoire (Load)	LI1	LI2	DI	LR	EX1	AM1	AM2	ER

La signification des différents étages est

- LI1 et LI2 : phases d'acquisition (lecture) de l'instruction
- DI : décodage de l'instruction
- LR : lecture des registres
- EX1 : calcul adresse mémoire, adresse de branchement, début exécution (EX1).
- EX2 : fin exécution (EX2) des instructions arithmétiques
- ER : écriture du résultat dans un registre
- AM1 et AM2 : phases accès cache données

Question 9 : Donner les latences entre instruction producteur et instruction consommateur en nombre de cycles pour les quatre cas ci-dessous en supposant que tous les bypass nécessaires

soient implémentés. (NB : la valeur n signifie que deux instructions dépendantes peuvent se suivre aux cycles i et i+n)

	Instruction producteur	Instruction consommateur	Latence
1	ADD R1,R2,R3	ADD R4, R1, R6	
2	SUB R1,R2,R3-	ST R1, 0 (R5)	
3	LDW R1, 4(R2)	XOR R8,R1,R7-	
4	LDW R1, 4(R2)	ST R1, 0 (R5)	

PREDICTION DE BRANCHEMENT

Question 10) Pour chacun des branchements suivants (considérés chacun séparément), proposer un type de prédicteur en justifiant parmi *Toujours pris / Prédicteur 1 bit / Prédicteur 2 bits*. Si plusieurs prédicteurs conviennent, on choisira celui qui utilise le moins de matériel.

- a) Branchement B1,
P,P,P,P,N,N,N,N,N,N,N,P,P,P,P,P,P,P,N,N,N,N,P,P,P,P.
- b) Branchement B2
P,P,P,N,P,P,P,P,N,P,P,P,P,N,N,N,N,P,N,N,N,N,P,N,N,N,N
- c) Branchement B3
P,P,P,P,N,P,P,P,P,N,P,P,P,P,P,P,N,P,P,P,P,P,N,P,P,P,P,P

ANNEXE : instructions et modes d'adressage ARM utilisés

LDR	Chargement d'un mot dans un registre	Rd ← Mem32 (adresse)
STR	Rangement d'un mot	Mem32 (adresse) ← Rd
LDRB	Chargement d'un octet non signé	Rd ← Zéro Mem8 (adresse)
LDRSB	Chargement d'un octet signé	Rd ← Extension signe Mem8 (adresse)
STRB	Rangement d'un octet	Mem8 (adresse) ← Rd
LDRH	Chargement 16 bits non signés	Rd ← Zéro Mem16 (adresse)
LDRSH	Chargement 16 bits signés	Rd ← Extension signe Mem16(adresse)
STRH	Rangement 16 bits	Mem16 (adresse) ← Rd
MV	Transfert	Rd ← Rs
SUBS	Soustraction + positionnement code cond.	Rd ← Rs1 - Rs2 ; Positionne CC
BGT	Branchement si plus grand	CP ← NCP + déplacement si GT = vrai

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