

TP3 : Architecture interne du contrôleur de cache

Jiang : 3602103

Li : 3770906

C) Application logicielle

Question C1 : **Quelle est l'adresse de la première instruction de la fonction main ? Quelle est l'adresse de la première instruction de la boucle (instruction correspondant à l'étiquette *loop*) ?**

main() est dans la première section de seg_code (app.ld) rangeant dans .mycode

->l'adresse de la première instruction de la fonction main :0x00400000 ;

Dans app.bin.txt :

l'adresse de la première instruction de la boucle : 0x004000010. (@main+16)

Question C2 : **Quelle sont les adresses de base des trois tableaux A, B, C ?**

Les tableau sont dans la première section de seg_data (app.ld) rangeant dans .mydata

l'adresse de base de tableau A : 0x01000080

l'adresse de base de tableau B : 0x01000100 (A+128)

l'adresse de base de tableau C : 0x01000180 (B+128)

Question C3 : **Pourquoi l'instruction *sw* est-elle placée après l'instruction *bne* (test de fin de boucle) ?**

Parce que le code est optimisé par le compilateur pour éviter le NOP. L'instruction après un branchement est toujours exécutée.

Question C4 : **Quel est le nombre de cycles nécessaires pour exécuter une itération de la boucle, dans l'hypothèse où toutes les instructions et toutes les données sont dans le cache (pas de MISS).**

7 cycles.(app.bin.txt)

D) Fonctionnement du cache instruction

Question D1 : **Sur combien de bits sont codés les trois champs BYTE, SET, et TAG d'une adresse 32 bits? Quelles sont leurs valeurs pour l'adresse 0x00400000 (correspondant à la première instruction du programme ?**

BYTE : 4 bits; SET : 3 bits; TAG : 25 bits;

leurs valeurs pour l'adresse 0x00400000 :

0000 0000 0100 0000 0000 0000 0 | 000 | 0000

TAG=0x8000 | SET | BYTE

Question D2 : **Représentez dans le tableau ci-dessous le contenu du cache instruction à la fin de la première itération de la boucle. Quelles instructions ont déclenché un MISS sur le cache instruction pour atteindre cet état ?**

TAG	V	WORD3	WORD2	WORD1	WORD0
0x8000	1	li	li	Addiu()	Lui(MISS)
0x8000	1	addi	addi	lw	lw(MISS)
0x8000	1	lui	sw	bne	add(MISS)

Les instructions de *lui*, *lw* et *add* ont déclenché un MISS sur le cache instruction.Les autres sont HIT.

Question D3 : Quel est le contenu du cache instruction à la fin de la 20e itération ? Calculez approximativement le taux de MISS lors de l'exécution complète du programme .

Le contenu du cache reste la même que la première itération.

A la fin de la 20e itération, il y a totalement 5 MISS (7 instructions après la boucle), le sum d'instructions égale $20 \times 7 + 4 = 144$, donc le taux de MISS $5/150 = 3.33\%$

Question D4 : Pour quel type de cache l'état MISS_SELECT est-il indispensable?

Pour la cache set-associative MISS_SELECT est indispensable.

Question D5 : Complétez le graphe de l'automate ICACHE_FSM ci-dessous, en attachant à chaque transition sa condition de franchissement.

A	IUNC·IREQ	L	VALID· ERROR	F	VALID·ERROR	J	VALID
B	IREQ·IMISS·IUNC	I	∅	G	VALID·ERROR	K	VALID·ERROR
C	IREQ+IMISS· IUNC·IREQ	N/M	∅	H	VALID	O	∅

Question D6 : Dans quel état être forcé cet automate lors de l'activation du signal RESETN? Quel doit être l'autre effet du signal RESETN sur le cache instruction ?

Dans l'état IDLE.

Le signal RESETN va aussi vider les valeurs de VALID et de ERROR.

E) Fonctionnement du cache de données

Question E1 : Quelles sont les valeurs des trois champs BYTE, SET, et TAG pour les adresses des éléments A[0] et B[0] des deux tableaux?

	Adresse	TAG	SET	BYTE
A[0]	0x100000080	0x020001	000	0000
B[0]	0x100000100	0x020002	000	0000

Donnez le contenu du cache de données à la fin de la première itération de la boucle, en précisant quelles instructions entraînent un MISS sur le cache de données et donc un gel du processeur.

TAG	V	WORD3	WORD2	WORD1	WORD0
0x020002	0	104	103	102	101
	0				

Dans la boucle loop, les deux premières lw entraînent un MISS sur le cache de données et donc un gel du processeur.

Question E2 : Calculez le taux de MISS sur le cache de données pour l'exécution complète des 20 itérations de cette boucle. Donnez le contenu du cache de données à la fin de la 20e itération de la boucle.

le taux de MISS=100%

TAG	V	WORD3	WORD2	WORD1	WORD0
0x010001 ₀	0	104	103	102	101
0x010001 ₁	0	108	107	106	105
0x010002 ₀	0	112	111	110	109
0x010002 ₁	0	116	115	114	113
0x010003 ₀	0	120	119	118	117
	0				
	0				
	0				

Question E3 : Complétez le graphe de cet automate, en attachant à chaque transition sa condition de franchissement.

A	DUNC·DREQ· \overline{WRITE}	L	VALID· \overline{ERROR}	F	VALID· \overline{ERROR}	J	\overline{VALID}
B	DREQ·DMISS· \overline{DUNC} · \overline{WRITE}	I	∅	G	VALID·ERROR	K	VALID·ERROR
C	$\overline{DREQ} + \overline{DMISS}$ · \overline{DUNC} ·DREQ· \overline{WOK}	N/M	∅	H	\overline{VALID}	O	∅
D	DREQ·WRITE· \overline{DMISS} ·WOK	E	DREQ·WRITE·DMISS·WOK	P	∅		

Question E4: Les expressions booléennes associées aux transitions de sortie de l'état WRITE_REQ sont très proches des expressions booléennes associées aux transitions de sortie de l'état IDLE. Quelle est la différence?

Dans la situation de $\overline{DREQ} + \overline{DMISS}$ · \overline{DUNC} ·DREQ· \overline{WOK} , l'état prochain de WRITE_REQ est IDLE mais pas lui-même.

F) Accès au PIBUS

Question F1 : Pourquoi les écritures ont-elles la priorité la plus élevée ? Quel est l'inconvénient de ce choix?

Parce que dans le write through, la capacité du tampon d'écriture est très petite, donc il faut écrire immédiatement. L'inconvénient : ça coûte cher et aussi moins efficace.

Question F2 : Quel est le mécanisme utilisé par les deux automates ICACHE_FSM et DCACHE_FSM pour transmettre une requête de lecture vers l'automate PIBUS_FSM?

On transmette la requête de lecture par signaux :

ICACHE_FSM : IUNC, IMISS

DCACHE_FSM : DUNC, DMISS

Comment le serveur signale-t-il aux clients que la requête a été prise en compte?

Le serveur le signale par r_pibus_ins, r_pibus_ins=true répond au ICACHE, r_pibus_ins=false est pour DCACHE.

Dans le cas d'une requête de lecture, comment le serveur signale-t-il au client que les données sont disponibles?

Quand `r_pibus_rsp_ok=true`, le client peut savoir que les données sont disponibles.

Question F3 : Pourquoi l'automate PIBUS_FSM n'a-t-il pas besoin de signaler qu'une requête d'écriture transmise par le tampon d'écritures postées s'est effectivement terminée?

Parce que l'écriture a la priorité et on va exécuter l'écriture toute de suite dans ce cas-là.

Quelle est l'utilité de la réponse dans le cas d'une transaction d'écriture?

Pour indiquer le cas d'erreur.

Question F4 : Complétez le graphe de l'automate PIBUS_FSM ci-dessous, en attachant à chaque transition sa condition de franchissement.

X	$R\bar{O}K+SC$	B'	$G\bar{N}T$	E	GNT	G	$LAST \cdot ACK[\bar{W}AIT]$
Y	$R\bar{O}K\bar{S}\bar{C}$ (IUNC+IMISS +DUNC+DMIS S)	C	\emptyset	E'	$G\bar{N}T$	G'	$L\bar{A}ST+ACK[WAIT]$
Z	$R\bar{O}K \cdot \bar{S}\bar{C} \cdot IUNC \cdot IMISS \cdot DUNC \cdot DMIS$	D	ACK[READY +ERROR]	F	LAST	H	ACK[READY+ERROR]
B	GNT	D'	ACK[WAIT]	F'	$L\bar{A}ST$	H'	ACK[WAIT]

Remarque : Dans `pibus_mips32_xcache.cpp`, dans le case `PIBUS_READ_DTAD`, il faut modifier le code à `if(p_ack.read() != WAIT)` pour le traitement d'erreur.

Question F5 : Calculez coût minimal du miss sur le cache instruction (nombre de cycles de gel du processeur en cas de miss).

10 cycles.

Pour cela, vous dessinerez explicitement le chronogramme décrivant pour chacun des deux automate, la succession des valeurs stockées dans les registres d'état. Même question pour le cache de données.

Question F6 : Calculez le nombre total de cycles pour exécuter les 20 itérations de la boucle de la section C, en prenant en compte les cycles de gel dûs aux miss sur le cache instruction et sur le cache de données. En déduire la valeur du CPI (nombre moyen de cycles par instruction).

$(7 \times 20 + 42 \times 10) / 140 = 4$

G) Expérimentation par simulation

Question G1 : A quel cycle le processeur exécute-t-il sa première instruction ? A quel code correspond cette instruction ? Quel est le coût d'un MISS sur le cache instruction (coût du MISS = nombre de cycles de gel du processeur).

Cycle 11

Question G2 : A quel cycle le processeur se branche-t-il au programme principal `main()` ?

Cycle 47

Question G3 : Quel est le coût d'un MISS sur le cache de données? A quel cycle le processeur termine-t-il l'exécution de la première itération de la boucle ? Quelle est la durée totale de la première itération ?

Cout de MISS = 10 cycles

La première itération se termine à cycle 107

Durée $107 - 61 = 46$ cycles.

Question G4 : Quelle est la durée de la seconde itération? Quelle est la durée de la troisième itération? Comment expliquez-vous que le coût du MISS pour les itérations 2 et 3 est plus élevé que pour la première itération?

2e itération durent 31 cycles.

Parce que MISS doit attendre l'écriture de l'itération précédent.

Question G5 : Quel est le taux de MISS sur le cache de données à la fin de l'exécution de la boucle? Quelle est la durée totale du programme main (sans compter le temps d'exécution de l'affichage du message final).

Dcache taux de MISS = 100%.

T = 650 cycles.

H) Optimisation

Pour cette application, les cycles de gel du processeur dûs aux MISS de conflit sur le cache de donnée représentent plus des 2/3 du temps d'exécution. Comment faut-il modifier les adresses des tableaux en mémoire pour minimiser le taux de MISS sur le cache de données, sans modifier l'architecture matérielle et sans modifier la structure du programme ? Recalculez la durée totale d'exécution du programme et la durée moyenne d'une itération pour cette nouvelle organisation de la mémoire, et vérifiez la justesse de vos estimations par simulation.

Base address de B = $16 * n$ bytes.