TP3 : Architecture interne du contrôleur de cache

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*) ?

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

l'adresse de la première instruction de la boucle : 0x0040000C. (@main+12)

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

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.

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

7 cycles.

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 :

BYTE : 0000 SET :000 TAG : 0000 0000 0100 0000 0000 0000 0

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
$0x004000_0$	1	lw	li	li	la
$0x004000_1$	1	add	addi	addi	lw
$0x004001_0$	1	addi	la	sw	bne

	0				
	0				
	0				
	0				
	0				

Les instructions de *la*, *lw* et *bne* ont déclenché un MISS sur le cache instruction.

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 complête 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 20×7+3+5+2=150, 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·$\overline{ERROR}$	F	VALID·$\overline{ERROR}$	J	$\overline{VALID}$
B	IREQ·IMISS·$\overline{IUNC}$	I	Ø	G	VALID·ERROR	K	VALID·ERROR
C	$\overline{IREQ}$+$\overline{IMISS}$·$\overline{IUNC}$ · $IREQ$	N/M	Ø	H	$\overline{VALID}$	O	Ø

Question D6 : Dans quel état ëtre 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?

	BYTE	SET	TAG
A[0]	0000	000	$0x010000_1$
B[0]	0000	000	$0x010001_0$

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
$0x010001_0$	0	104	103	102	101
	0				
	0				
	0				
	0				
	0				
	0				
	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	Ø	G	VALID·ERROR	K	VALID·ERROR
C	$\overline{DREQ}$+$\overline{DMISS}$·$\overline{DUNC}$·$DREQ$·$\overline{WOK}$	N/M	Ø	H	$\overline{VALID}$	O	Ø
D	DREQ·WRITE·$\overline{DMISS}$·WOK	E	DREQ·WRITE·DMISS·WOK	P	Ø		

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 $\overline{DREQ}+\overline{DMISS}\cdot\overline{DUNC}\cdot DREQ\cdot\overline{WOK}$, 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 coût 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 transmettre 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	ROK+SC	B'	$\overline{GNT}$	E	GNT	G	LAST·$\overline{ACK[WAIT]}$
Y	$\overline{ROK}\ \overline{SC}$(IUNC+ IMISS+DUNC +DMISS)	C	$\emptyset$	E'	$\overline{GNT}$	G'	$\overline{LAST}$+ACK[WAIT]

Z	$\overline{ROK} \cdot \overline{SC} \cdot \overline{IUNC} \cdot \overline{IMISS} \cdot \overline{DUNC} \cdot \overline{DMISS}$	D	ACK[READY +ERROR]	F	LAST	H	ACK[READY+ERROR]
B	GNT	D'	ACK[WAIT]	F'	$\overline{LAST}$	H'	ACK[WAIT]

->Remarque : Dans pibus_mips32_xcache.cpp, dans le case PIBUS_READ_DTAD, il faut modifer 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).

9 cycles.

Pour cela, vous dessinerez explicitement le chronogramme décrivant pour chacun des deux automate, la succession des valeurs stockées dans les registres d'etat. 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.

(150+18×20+5×9)/150=3.7

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).**

1)A cycle 56 le processeur exécute sa première instruction. On peut voir que a cycle 55 on a déjà lu les 4 premières instruction dans la cache , car l'état du processeur est de ICACHE_MISS_UPDT(cycle 55) à ICACHE_IDLE. (cycle 56)

2)Il correspond au code : *la $8, A .*

3)Le MISS est de cycle 46 à cycle 56, donc il y a 10 cycles de gel du processeur.

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

A cycle 46 le processeur se branche au programme principal main().

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 ?

1)Le coût d'un MISS sui DCACHE= 10 cycles. (cycle 71 à cycle 81) ;

2)A cycle 105 la première itération de la boucle est terminée. Et il commence à cycle 71, donc la durée totale de la première itération est 105-71+1=35 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?

1)La deuxième itération commence à cycle 106 et termine à cycle 136, donc la durée de la secondes itération=136-106+1=31 cycles.

2) La troisième itération commence à cycle 137 et termine à cycle 167, donc la durée de la secondes itération=167-137+1=31 cycles.

Question G5 : Quesl 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).

Le taux de MISS sur DCACHE=100% ;

Le début du main est à cycle 46 et le fin est à cycle 725. La durée du programme main=725-46=680 cycles.

H) Optimisation

Pour cette application, il y a MISS de conflit sur le DCACHE qui est embêtant. Pour optimiser, il faut modifier les adresses des tableaux en mémoire pour minimiser le taux de MISS sur le DCACHE. Pour cela , on modifie la valeur du *.space* de A et C de 48 à 64.