Cours "Architecture des Systèmes Multi-Processeurs"

TP4 : Caractérisation et dimensionnement des caches

C) Système mémoire presque parfait

Question C1 : Lancez la simulation, avec des caches très grands : 256 sets, 16 mots par ligne, et 4 niveaux d'associativité soit une capacité de 64 Koctets pour chacun des deux caches. On choisit également un tampon d'écritures postées de profondeur 8 mots de 32 bits. Quel est le temps d'exécution de l'application?

75725 cycles.

Question C2 : Relancez la simulation, et déterminez les taux de MISS pour le cache instruction et le cache de données ainsi que la valeur du CPI. Les taux de MISS sont-il constants au cours du temps ? Comment évoluent-t-ils au cours des 1000 premiers cycles? Interprêtez ce comportement.

CPI = 1.14 IMISS RATE = 0.0005 DMISS RATE = 0.0015

Au debut démarage de la machine, les caches () sont vides. Donc il y a plus de MISS. Une plus grande cache a moin de IMISS à la fin d'execution.

Question C3 : En analysant le contenu des fichiers pibus_mips32_xcache.cpp et pibus_mips32_xcache.h, expliquez comment sont calculés le CPI et les deux taux de MISS.

run_cycles = c_total_cycles - c_frz_cycles

INSTRUCTIONS = run_cycles

CPI =c_total_cycles / run_cycles

CACHED READ RATE = (c_dread_count – c_dunc_count) / run_cycles

UNCACHED READ RATE =c_dunc_count / run_cycles

WRITE RATE = c_write_count / run_cycles

IMISS RATE = c_imiss_count / run_cycles

DMISS RATE = c_dmiss_count / (c_dread_count - c_dunc_count)

IMISS COST = c_imiss_frz / c_imiss_count

DMISS COST = c_dmiss_frz / c_dmiss_count

UNC COST =c_dunc_frz / c_dunc_count

WRITE COST = c_write_frz / c_write_count

D) Influence de la capacité du cache instruction

Question D1 : Relevez, pour chaque configuration, la durée d'exécution du programme, le taux de MISS sur le cache instruction, le cout du miss instruction et la valeur du CPI. Comment interprêtez-vous ces résultats ?

Si SETS diminue (et IWORDS ,IWAYS ne changent pas), c'est-à-dire la capacité de la cache augmentent. La durée, MISS Rate, et CPI augmentent aussi. Mais MISS Cost dimiue.

ISETS	Cycles	IMISS Cost	IMISS Rate
256	75725	24.8485	0.00102235
64	94253	24.8485	0.0224879
16	150783	24.3445	0.0964379
4	170355	23.8903	0.125334

1	238625	23.3046	0.242697

Question D2 : Pourquoi la valeur obtenue pour le coût du MISS est-elle différente de la valeur estimée dans le TP3 ? Comment expliquez-vous que le coût du miss puisse avoir une valeur non entière? Pourquoi tous lesMISS nont-ils pas le même coût?

Le cache transfére par ligne (lecture). Le capacité de ligne est augmentée à 8 (IWORDS = 8).

Le temps entre le detection MISS et l'accès au BUS n'est pas fixés. Parce que il'y a plusieur CPU et t L'écriture est toujours à priorité. Traitement de MISS doit attendre pour l'écriture.

E) Influence de la largeur de la ligne de cache

Question E1 : Représenter graphiquement la durée d'exécution du programme en fonction de la largeur de la ligne de cache. Quelle est la configuration la plus efficace ? Comment expliquez-vous ce résultat ?

IWAYS = 1 DSETS=256 DWORDS=16 DWAYS=4 Capacity = 8K

ISETS	IWORDS	Cycles	CPI	IMISS Cost	IMISS Rate	IMISS rate*cost
256	1	177661	4.23789	7.18335	0.354659	2.5476
128	2	141032	2.89528	8.54846	0.165856	1.4178
64	4	124500	2.382	11.3416	0.0865364	0.98146
32	8	118708	2.19817	16.1087	0.0516082	0.83134
16	16	117885	2.14317	24.3445	0.0324516	0.79002
8	32	140825	2.55437	39.9283	0.030092	1.2015

La largeur de ligne augmente, IMISS Cost augmente, IMSS Rate diminue. Dans ce cas, quand ISETS=16 et IWORDS=16 , le coût est mion cher.

F) Influence de la capacité du cache de données

Question F1 : Relevez, pour chaque valeur, la durée d'exécution du programme, le taux de MISS sur le cache de données, le cout du miss de données et la valeur du CPI.

DSETS	Cycles	CPI	DMISS Cost	DMISS Rate
256	75796	1.33545	16.2093	0.00266485
64	76332	1.34489	17.274	0.00451565
16	101807	1.79373	17.1067	0.0905854
4	153010	2.69588	16.4671	0.232168
1	244871	4.31438	15.9252	0.403676

Quand la capacité diminue, DMISS Cost diminue, mais DMISS Rate augmente.

G) Influence de la profondeur du tampon d'écritures postées

On étudie enfin l'influence de la profondeur du tampon d'écritures postées. Le paramètre WBUF_DEPTH définit le nombre maximum de requêtes d'écritures (correspondant à des instructions de type sw ou sb) qui peuvent être stockées dans ce tampon.

Question G1.1 : Comment est réalisé le tampon d'écriture postées?

Le buffer est un FIFO pour l'écriture(DCACHE_FSM : WRITE_REQ ou BCU : Write).

Question G1.2 : quelle sont les informations qu'il faut stocker dans le tampon pour chaque requête d'écriture enregistrée ?

| adresse | DATA | Size |

Question G1.3 : Que se passe-t-il lorsque le processeur effectue une requête d'écriture alors que le tampon décriture postées est plein?

Le processus est gelé.

Question G1.2 : Que se passe-t-il lorsque le processeur fait une requête de lecture (instruction ou donnée) qui fait miss, alors que le tampon d'écriture est non-vide ? Pourquoi ce comportement ?

La traitement de MISS doit attendre jusqu'à l'ecriture est effectuée.

Parce que l'écriture est à priorité et le buffer doit être traité d'abord.

Question G2 : Mesurez le temps d'exécution de l'application, le CPI, le coût des écriture, et La fréquence des écritures pour des profondeurs de 1, 2, 4, et 8 mots. A quoi correspond le coût des écritures? Comment expliquez-vous le fait que le coût des écritures soit si faible?

BUF	Cycles	CPI	DMISS Cost	DMISS Rate	IMISS COST	IMISS RATE	WRITE COST
8	75725	1.3342	24.32	0.00155106	24.8485	0.000581426	0
4	75725	1.3342	24.32	0.00155106	24.8485	0.000581426	0
2	76619	1.34995	24.08	0.00155106	24.8485	0.000581426	0.118504
1	78847	1.3892	23.44	0.00155106	24.5152	0.000581426	0.463371

Dans pibus_mips32_xcache:

c_write_frz = nombre de fois quand le buffer est plein

WRITE COST = c_write_frz / c_write_count

WRITE RATE = c_write_count / run_cycles

La largeur du buffer est grande, il est jamais plein.