TP5 : Partage du bus dans les architectures multi-processeurs

JIANG Hongbo 3602103

B) Architecture matérielle

Question B1: En consultant l'en-tête du chier *pibus_frame_bu er.h*, précisez la signi cation des arguments du constructeur du composant PibusFrameBu er.

Name :nom d'instance
segtab :table de segments
width: 256
subsampling : =420

Question B2: quelle est la longueur du segment associé à ce composant ?

256*256 (segsize<width*height).

C) Compilation de l'application logicielle

tgtid : l'index de cible latency : délais

height : 256

Question C1 : **Pourquoi faut-il utiliser un appel système pour accéder (en lecture comme en écriture) au contrôleur de frame-bu er? Que se passe-t-il si un programme utilisateur essaie de lire ou d'écrire directement dans le Frame Bu er, sans passer par un appel système?**

Parce qu'il faut accéder au contrôleur de frame-bu er via driver. Si on écrire directement dans le Frame Bu er , on risque à détruire le matériel et le système d'exploitation ne va pas le permettre.

Question C2 : **Pourquoi préfère-t-on construire une ligne complête dans un tableau intermédiaire de 256 pixels plutôt que d'écrire directement l'image - pixel par pixel - dans le frame bu er?**

Parce que le longeur du segment associé à le frame bu er est 256*256, c'est plus pratique de construire une ligne complête dans un tableau pour augmenter la vitesse.

Question C3 : Consultez le chier *stdio.c*, et expliquez la signi cation des trois arguments de l'appel système *fb_sync_write()*. Complétez le chier *main.c* en conséquence.

Des trois arguments de l'appel système *fb_sync_write()* : *o set, bu er, length*. O set : line*256, indiquer l'endroit de départ
Bu er : buf[pixel], pour stocker les données.
Length : 256,la taille de chaque ligne qu'on veut écrire dans le bu er.

D) Caractérisation de l'application logicielle

Question D1 : **Quel est le nombre de cycles nécessaires pour a cher l'image avec un seul processeur. Combien d'instructions ont été exécutées pour a cher l'image? Quel est le nombre moyen de cycles par instruction (CPI) ?**

le nombre de cycles nécessaires pour a cher l'image avec un seul processeur = **5012208** cycles.

#Instruction=**4046323** CPI=**1.2387**

**Question D2 : Quel est le pourcentage d'écritures sur l'ensemble des instructions exécutées ? Quel est le pourcentage de lectures de données ? Quels sont les taux de miss sur le cache instruction et sur le cache de
données ? Quel est le coût d'un miss sur le cache instructions ? Quel est le coût d'un miss sur le cache de données ? Quel est le coût d'une écriture pour le processeur ? On rappelle que les couts se mesurent en nombre moyen de cycles de gel du processeur. Comment expliquez-vous que ces coûts ont des valeurs non entières?**

Parce que le BUS peut être occupé parfois, qui produit le con ict et augmente le coût de IMISS, et comme c'est une valeur moyenne et le con ict n'existe pas toujours, donc le coût du miss peut être une valeur non entière.

Question D3 : Evaluez le nombre de transactions de chaque type pour cette application. Que remarquez-vous ?

#transactions= #instructions * taux de miss. On peut remarquer qu'il y a plus de MISS sur DCACHE.

E) Exécution sur architecture multi-processeurs

Question E1 : Modi ez la boucle principale dans le chier main.c pour partager le travail entre les di érents processeurs de l'architecture.

Dans le chier **main.c**, on ajoute le code : if(line % nprocs ==n) pour que chaque processeurs traite les di érentes lignes de l'image.

Question E2 : Pourquoi les piles d'exécution des N programmes s'exécutant sur les N processeurs doivent-elles être strictement disjointes? Modi ez le chier reset.s, pour initialiser le pointeur de pile à une valeur dépendant du numéro du processeur, de telle sorte que chaque tâche possède une pile de 64 Koctets, quelque soit le nombre de processeurs (compris entre 1 et 8).

Dans reset.s :

lui $28, 0x0001

mult $28, $27

mflo $27

addu $29, $29, $28

addu $29, $29, $27

#stack size = 64 Koctets

multi processeur

Question E3 : Donnez deux raisons pour lesquelles le code binaire doit être recompilé chaque fois qu'on change le nombre de processeurs.

Parce qu'on a modi é le nombre de processeur dans les chiers *con g.h* et *tp5_top.cpp*.

Question E4 Remplissez le tableau ci-dessous, et représentez graphiquement speedup(N) en fonction de N.

On a noté la valeur la plus grande parmi plusieurs processeurs parce qu'il faut a endre jusqu'à tous les processeurs ont ni le travail.

Comment expliquez-vous que le speedup ne soit pas linéaire?

Parce que le plus le nombre de processeurs est élevé, le plus possible que le bus est chargé. C'est la saturation du bus.

F) Evaluation des temps d’accès au bus

Question F1 Pourquoi faut-il absolument e ectuer la mesure au moment précis où l'application se termine?

Parce que le nombre de cycles est énorme.

Question F2 Remplissez le tableau ci-dessous, en exécutant successivement le programme sur di érentes architectures:

Question F3 : Interprétez ces résultats. La dégradation de la valeur du CPI quand on augmente le nombre de processeurs est-elle principalement dûe à l'augmentation du coût des MISS, ou à l'augmentation du coût des écritures?

C'est principalement à cause de l'augmentation du coût des écritures parce que le taux d'écriture est beaucoup plus élevé que le taux de MISS.

G) Modélisation du comportement du bus

Question G2 : En exploitant les résultats de la partie D (pourcentage de lectures cachées, pourcentage d' écritures, taux de MISS sur les caches, valeur du CPI), calculez la fréquence de chacun des 4 types de transaction et complétez le tableau ci-dessous. Puisque notre unité de temps est le cycle, ces fréquences seront exprimées en *nombre d'événements par cycle*.

Question G3 : **En utilisant les résultats des deux questions précédentes, calculez le pourcentage de la bande passante du bus utilisé par un seul processeur. En déduire le nombre de processeurs au delà duquel le bus commence à saturer.**

Le pourcentage de la bande passante du bus = 0,3560 nombre de processeurs saturé = 2,809