TP8 : Contrôleur de disque & partage de périphériques

XU Weiqin 3410681

HUANG Xingyun 3602284

B) Contrôleur de disque

Question B1 : Quelle est la signification des argument *block_size* et *latency* du constructeur du composant *PibusBlockDevice* ?

block_size : la taille de chaque bloc, c'est-à-dire la nombre d'octets dans un bloc : (128/256/512/1024)

latency : latency d'accès (cycles)

Question B2 : Combien de blocs une image occupe-t-elle sur le disque pour des blocs de 512 octets?

672.

Question B3 : Quels sont les registres adressables du contrôleur de disque, et quel effet a une écriture ou une lecture dans chacun de ces registres?

BLOCK_DEVICE_BUFFER/ BLOCK_DEVICE_LBA/ BLOCK_DEVICE_COUNT/ BLOCK_DEVICE_OP / BLOCK_DEVICE_STATUS / BLOCK_DEVICE_IRQEN / BLOCK_DEVICE_SIZE / BLOCK_DEVICE_BLOCK

Quand on est dans les états de READ_ERROR, READ_SUCCESS, WRITE_ERROR, WRITE_SUCCESS, une lecture accès à BLOCK_DEVICE_STATUS va reset le master FSM à l'état IDLE ;
Des écritures accès à les registres BUFFER, COUNT, LBA, OP seraient ignoré.

Question B4 : Quels sont les différentes valeurs de l'état interne du contrôleur de disque qui peuvent être lues par le logiciel, et quelle est la signification de chacun de ces états?

Les états internes : BLOCK_DEVICE_IDLE, BLOCK_DEVICE_BUSY

BLOCK_DEVICE_READ SUCCESS est de la valeur 2 (C'est-à-dire la lecture accès à la disque est fini.) BLOCK_DEVICE_READ ERROR est de la valeur 4 (C'est-à-dire qu'il y a des exceptions pendant le transfert d'information)
BLOCK_DEVICE_WRITE SUCCESS 3 ;
BLOCK_DEVICE_WRITE ERROR 5

C) Architecture matérielle

Question C1 : Pourquoi l'utilisation du composant *PibusBlockDevice* impose-t-elle d'augmenter la valeur du time-out du composant *PibusSegBcu*? Quelle valeur faut-il donner au paramètre *timeout* du constructeur du composant *PibusSegBcu*?

Le composant matériel *PibusBlockDevice* permet d'accéder à un dispositif de stockage externe, mais des temps d'accès sont souvent très grands, donc il demander une valeur plus grande.

Le paramètre de timeout prend la valeur de latency (:1000 cycles) du composant PibusBlockDevice.

Question C2 : Quelles sont les valeurs de l'adresse de base et de la longueur du segment associé au contrôleur de disque (IOC).

0x92000000 ;

Compte-tenu du nombre variable de processeurs dans cette architecture, quelle sont les longueurs des segments associés aux composants ICU, TTY et TIMER?

ICU : 32*4procs = 128 octets ;
TTY : 16*4procs = 64 octets ;
TIMER : 16*4procs = 64 octets.

Question C3 : Combien y-a-t-il de composants *maitres* dans cette architecture? Combien de composants *cibles*?

Maitres : 6 (4 processeurs + DMA + IOC)
Cibles : 7 (ROM, RAM, TTY, FBF, TIMER, IOC, DMA)

Question C4 : Compte-tenu du nombre variables de processeurs dans cette architecture, combien le composant ICU reçoit-il de lignes d'interruption entrantes, en provenance des 4 périphériques TTY, TIMER, DMA et IOC?

Proccesur 0 a 4 lignes d'interruption entrantes, pour les autres 3 processeurs il y en a 2 chacun.
4+2*3=10 lignes d'interruption entrantes.

Combien possède-t-il de lignes d'interruptions sortantes? Comment les IRQs provenant des périphériques sont-elles connectées sur les ports IRQ_IN[i] du composant ICU ?

D) Code de boot

Question D1 : Rappelez pourquoi - en l'absence d'un mécanisme de mémoire virtuelle - l'initialisation du pointeur de pile dépend du numéro de processeur.

Parce que chaque processeur possède son propre pile, il faut les initialiser séparément.

Question D2 : Rappelez le mécanisme général qui permet au système d'exploitation de router - par logiciel - les différentes lignes d'interruption entrantes sur le composant ICU vers différents processeurs.

Question D3 : Dans le cas d'une architecture à 4 processeurs, quelles sont les valeurs à stocker dans les 4 registres de masque de l'ICU si on veut réaliser le routage suivant:

- **IRQ_TIMER[0], IRQ_TTY[0] IRQ_DMA et IRQ_IOC vers le processeur 0**
- **IRQ_TIMER[1], IRQ_TTY[1] vers le processeur 1**
- **IRQ_TIMER[2], IRQ_TTY[2] vers le processeur 2**
- **IRQ_TIMER[3], IRQ_TTY[3] vers le processeur 3**

Processeur 0 : 0xF
Processeur 1 :0x30
Processeur 2 :0xC0
Processeur 3 :0x300

E) Application logicielle de traitement d'image

Question E1 : Quels sont les arguments de l'appel système *ioc_read()*?

Arguments :
lba : l'index du premier bloc dans la disque ;
buffer : l'adresse de base du buffer de mémoire ;
count : le nombre des blocs à transfer.

Que fait cet appel système?

Cet appel système permet de transfer les données d'une image sur le disque, et copie de cette image dans un tampon mémoire *buf_in*. La destination de mémoire buffer doit être dans l'espace user addresse.

Cet appel système attend-il que le transfert soit terminé pour rendre la main? Dans quel cas cet appel système est-il bloquant?

Question E2 : Que fait l'appel système *ioc_completed()*?

Cet appel système détecte la completion d'un I/O transfert et renvoie >0 s'il y a des erreurs et renvoie 0 si c'est un succès.

Quels sont ses arguments? Pas d'argument.

Cet appel système est-il bloquant? C'est bloquant, le processeur sera bloqué jusqu'à la prochaine interruption.

Question E3: Quel problème observez-vous lors de l'affichage des images suivantes? Comment expliquez-vous ce dysfonctionnement ?

On ne passe pas à l'image suivante après avoir appuyé sur une touche du clavier.

Parce que le contenu du cache reste le même sans snoop.

Question E4: Quelles conditions font-elles sortir l'automate SNOOP_FSM de l'état IDLE ?

7 conditions :

1) SNOOP => SNOOP_INVAL_* ou SNOOP_FLUSH (pendant un cycle) => IDLE
2) CACHED READ MISS => MISS_SELECT => MISS_WAIT => MISS_UPDT => IDLE
3) UNCACHED READ => UNC_WAIT => UNC_GO => IDLE
4) WRITE HIT => WRITE_UPDT => WRITE REQ
5) WRITE MISS => WRITE_REQ
6) SC => SC_WAIT => IDLE
7) XTN INVAL => INVAL (pendant un cycle) => IDLE

La stratégie mise en oeuvre en cas de hit externe est-elle une mise à jour ou une invalidation?

Le premier hit externen est une invalidation est les suivants est les mises à jour.

Question E5: Pourquoi la détection de plusieurs hit externes consécutifs pose-t-elle un problème particulier? Comment ce problème est-il résolu?

Quand on cherche des informations de disque, on les prend dans la disque et on les conserve dans le mémoire buffer et cette adresse est fixée. S'il y a des hits externes consécutifs, CPU suppose qu'il y a déjà les informations demandées dans le cache et prend le même contenu.

Pour résoudre ce problème, il faut activer la fonction snoop pour détecter le changement des données.

Question E6: Mesurez, pour les 4 premières images, les durées d'exécution de chacune des trois étapes du programme (chargement, seuillage, affichage) et reportez ces mesures dans un tableau.

	chargement	seuillage	affichage
Image 0	41792	439202	120424
Image 1	40533	439355	120139
Image 2	40587	439382	120139
Image 3	40587	439382	120139

F) Exécution sur architecture multi-processeurs

Question F1 : Sur les trois phases de traitement (chargement, filtrage, affichage), lesquelles vont effectivement pouvoir être parallélisées?

La phase filtrage peut être parallélisée parce qu'on ne demande pas l'utilisation du bus dans cette phase.

Question F3 : Analysez en détail le code de la fonction *_ioc_get_lock()* que vous trouverez dans le fichier drivers.c, et expliquez ce que fait ce code).

C’est une fonction qui est utilisé par les fonctions _ioc_read et _ioc_write pour verouiller IOC en utilisant les instructions LL/SC.

Question F4 : Quelle est la fonction système qui libère le verrou protégeant l'accès exclusif au composant IOC. Pourquoi n'est-il pas nécessaire d'utiliser une instruction particulière pour libérer le verrou?

La fonction _ioc_complete qui rend la variable _ioc_done à zéro et libère le verrou. C’est fait quand la complétion est signalé, donc il n’est pas nécessaire d’utiliser une instruction particulière pour libérer le verrou.

G) Réalisation matérielle du LL/SC

Question G1 : Pour quelle raison réalise-t-on cet enregistrement du côté du processeur plutôt que du côté de la mémoire?

Parce que c’est plus rapide d’accéder aux données si on l’enregistre du côté du processeur.

Question G2 : Dans le scénario décrit ci-dessus, comment le processeur P0 est-il informé de l'écriture effectuée par P1?

Question G3 : Pour vérifier votre hypothèse, exécutez l'application de traitement d'image en désactivant le mécanisme de snoop, grâce à l'argument (-SNOOP 0) sur la ligne de commande. Comment expliquez-vous ce que vous observez ?

Question G4 : Résumez en une phrase les deux utilisations du mécanisme de snoop qui on été mis en évidence dans ce TP.

Le snoop surveille :

1) La modification Cache fait à mémoire ;
2) La modification Cache fait à mémoire ;