Nom : AIDER
Prénom : Smail
N°Etudiant : 3603379
Parcours : SAR
Responsable : M Alain Greiner

Compte-Rendu TP5
Partage du bus dans les architectures multi-processeurs

B) *Architecture matérielle*

Question B1

Signification des arguments du constructeur du composant PibusFrameBuffer :
- name : le nom de l'instance.
- index : le numéro de la cible
- segmap : l'adresse de la table des segments
- latency : nombre de cycles d'attente
- width : nombre de pixels par ligne
- height : nombre de lignes qui composent une image
- subsampling : format d'un pixel

Question B2

La longueur du segment associé au composant PibusFrameBuffer est de :
- width * height (FB_NPIXEL*FB_NLINE), 256 * 256 = 64Ko.

C) *Compilation de l'application logicielle*

Question C1

Comme tout les périphériques, l'accès au contrôleur de frame-buffer doit se faire via un appel système pour pouvoir vérifier les droits d'accès. De plus, le segment associé à se périphérique (FBF) se trouve dans l'espace kernel (SEG_FBF_BASE 0x96000000), et tout accès à cette zone doit se faire via un appel système.

Si un programme utilisateur essaie de lire ou écrire directement dans le Frame Buffer, la garantie d'intégrité et de confidentialité de données n'est plus respectée.

Question C2

On préfère construire une ligne complète dans un tableau intermédiaire plutôt que d'écrire - pixel par pixel - dans le frame buffer pour réduire le nombre de requêtes émises sur le BCU (Pibus controler) parce que l'accès à ce dernier est couteux (sort de rafale).

Question C3

Arguments de l'appel système « fb_sync_write() » :
- OFFSET : la position du premier octet de donnée à envoyer dans le Frame Buffer.
- BUFFER : l'adresse de base du buffer dans l'espace utilisateur.
- LENGTH : la taille du buffer à envoyer.

D) *Caractérisation de l'application logicielle*

Question D1

- Le nombre de cycles nécessaire pour afficher l'image est de 5705100 cycles.
- Le nombre d'instructions exécutées : 4013106
- Le nombre moyen de cycles par instruction (CPI) : 1.42162

Question D2

- WRITE RATE : 0.142771
- DREAD RATE : 0.266446 (CACHED) + 0.0013122 (UNCACHED) = 0,2677582
- IMISS RATE : 0.00887293
- DMISS RATE : 0.00911551
- IMISS COST : 15.9823
- DMISS COST : 14.4224
- WRITE COST : 0

Les couts ont une valeurs non entières car le temps d'attente entre la détection du MISS et l'accès au bus est variable.

Question D3

On comparant le WRITE_RATE avec le DREAD_RATE, on peut dire qu'il y a à peu prés 2 fois plus de lectures que d'écritures de données.

E) *Exécution sur architecture multi-processeurs*

Question E1

Nouveau **main.c** :

```
for(line = 0 ; line < NLINE ; line++)
{
    if((line % nprocs) == n){
            for(pixel = 0 ; pixel < NPIXEL ; pixel++)
            {
                buf[pixel] = build(pixel, line, 5);
            }
            if ( fb_sync_write(line*NPIXEL, buf, NPIXEL) )
                tty_printf(" !!! wrong transfer to frame buffer for line %d\n", line);
            else
                tty_printf(" - building line %d\n", line);
    }
}
```

Question E2

Les piles d'exécution des N programmes s'exécutant sur les N processeurs doivent être strictement disjointes car chaque programmes possèdent ses propres données (lignes de l'image).

Nouveau **reset.s** :

```
# initializes stack pointer
        mfc0    $27,    $15,    1               # $27 <= proc_id
        la      $29,    seg_stack_base
        li      $26,    0x10000                 # $26 <= 64 Kbytes
        mul     $27,    $26,    $27             # $27 <= 64 * proc_id
        addu    $29,    $29,    $27             # @base for proc_id
        addu    $29,    $29,    $26             # stack size = 64 Kbytes
```

Question E3

Il faut recompiler à chaque changement du nombre de processeurs car :
- On touche au fichier de configuration « config.h »
-

Question E4

speedup(N) = Temps de calcul sur 1 processeur / Temps de calcul sur N processeurs.

	1 proc	2 proc	4 proc	6 proc	8 proc
cycles	5705100	2879479	1596060	1549722	1560505
speedup	1.0	1.981	3.574	3.681	3.655

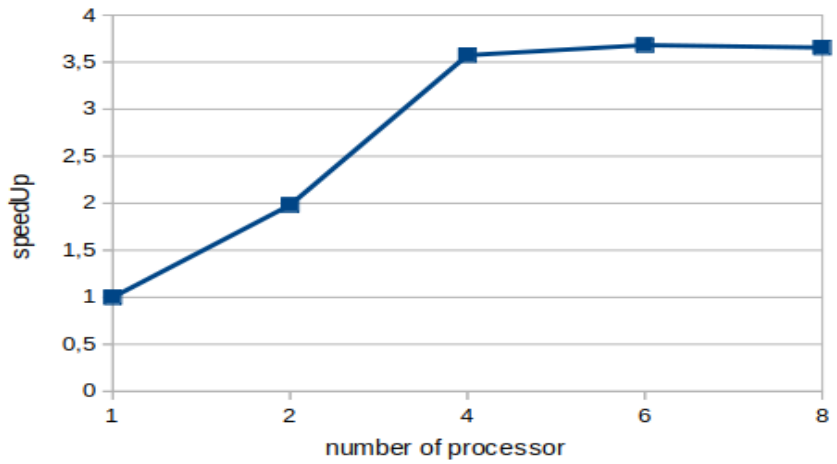

Figure 1: Accélération liée à la parallélisation sur N processeurs

Le speedup n'est pas linéaire car au bout de 4 processeurs, l'accélération commence à se dégrader due à l'augmentation du temps d'accès au bus.
Le nombre de cœurs n'est pas proportionnelle pas à la quantité de données passer sur le bus car sur ce type d'architecture (pibus), on a au plus 1 transaction/cycle.

F) *Évaluation des temps d'accès au bus*

Question F1

Il faut effectuer la mesure au moment précis ou l'application se termine car au-delà de cette instant, le(s) processeur(s) continue(nt) comme même de tourner, ce qui va fausser les résultats.

Question F2

NPROCS	*1*	*2*	*4*	*6*	*8*
IMISS COST	15.9823	17.4854	23.5733	43.5437	62.0385
DMISS COST	14.4224	16.4209	21.2969	36.656	51.3367
WRITE COST	0	0	0.570697	3.84246	7.70526
ACCESS_TIME	1	1.45955	4.90435	10.7303	15.816
CPI	1.42162	1.43298	1.58527	2.28687	3.07555

IMISS RATE	0.0088729	0.008484	0.0084803	0.0084924	0.0086974
DMISS RATE	0.0091155	0.008175	0.0081925	0.0084693	0.0088651
WRITE RATE	0.142771	0.142638	0.142482	0.142393	0.142202

Question F3

On observe que les couts de MISS augmentent dès lors que le nombre de processeurs augmente. Cela est due à l'augmentation du nombre de cycles de gel. Rappelant que :
- I/DMISS_COST = I/DMISS_FRZ / I/DMISS_COUNT
On exécutant le programme sur plusieurs processeurs, le nombre de miss instruction/donnée diminue, car il y a moins de lignes à traiter (265/N).
Donc, cette augmentation (MISS_COST) est proportionnelle au nombre de cycles de gel (MISS_FRZ) qui est lui même proportionnelle au temps d'accès au bus.

Tant qu'aux taux de miss et d'écritures, ils restent plutôt constant.

La dégradation de la valeur de CPI est due principalement ?

G) *Exécution sur architecture multi-processeurs*

Question G1

Calcule du nombre de cycles d'occupation du bus pour les 4 types de transaction :
L'automate PIBUS_FSM peut générer six types de transactions sur le bus correspondant au six événement suivants :

1. Le tampon WBUF est non-vide, et demande une transaction de type WRITE (écriture d'un mot)
2. l'automate DCACHE_FSM demande une transaction de type SC (écriture conditionnelle d'un mot)
3. l'automate DCACHE_FSM demande une transaction de type DUNC (lecture d'un mot non cachable)
4. l'automate DCACHE_FSM demande une transaction de type DMISS (lecture d'une ligne de cache)
5. l'automate ICACHE_FSM demande une transaction de type IUNC (lecture d'un mot non cachable)
6. l'automate ICACHE_FSM demande une transaction de type IMISS (lecture d'une ligne de cache)

- IMISS : 10 cycles
→ 1 cycle dans l'état « AD » : [A0]
→ 7 cycles dans l'état « DTAD » : [A1:A7, D0:D6]
→ 1 cycle dans l'état « DT » : [D7]
→ 1 cycle mort
- DMISS : 10 cycles
→ idem que IMISS
- UNC : 3 cycles
→ 1 cycle dans l'état « AD » : [A0]
→ 1 cycle dans l'état « DT » : [D0]
→ 1 cycle mort
- WRITE : 3 cycles
→ 1 cycle dans l'état « AD » : [A0]
→ 1 cycle dans l'état « DT » : [D0]
→ 1 cycle mort

Total : 26 cycles.

Question G2

	Temps_Occupation	Fréquence
IMISS	10	0,006241422
DMISS	10	0,006412058
UNC	3	0,000923031
WRITE	3	0,100428385

- Fréquence :
(nombre d'événement / cycle) = (nombre d'événement / instruction) * (nombre d'instruction / cycle)
→ le nombre d'instruction / cycle = nombre d'instruction / nombre de cycle = 1 / CPI.
→ le nombre d'événement / instruction = le taux d'un événement.

Question G3

Le pourcentage de la bande passante du bus utilisé par un seul processeur est de :

- 10 * (0,006241422 + 0,006412058) + 3 * (0,000923031 + 0,100428385) = 0,430
→ 43% /processeur.
Donc le bus commence à saturer au-delà de 2 processeurs donc : 3. CPU.