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

SORBONNE UNIVERSITÉ

Compte-Rendu TP6
Interruptions vectorisées / communication avec les périphériques

C) *Composants périphériques*

Question C1

Le composant **PibusMultiTimer** est une cible car il possède ces propres registres, il n'a donc pas besoin d'accéder à la mémoire. On dit qu'il n'a pas une « Capacité DMA ».
L'argument « ntimer » du constructeur sert à définir le nombre de timers (jusqu'à 32 timers) qui gênèrent des interruptions périodiques, programmables par logiciel.

Les registres adressables par ce composant :
- TIMER_VALUE[i] @(0x0) ; sert à stocker la valeur courante du timer.
- TIMER_RUNNING[i] @(0x4) ; booléen qui indique s'il y a une interruption (IRQ[i]) activée, si le cas, décrémenter le compteur TIMER_COUNT[i] et activer IRQ[i].
- TIMER_PERIOD[i] @(0x8) ; sert à indiquer la période entre deux interruptions.
- TIMER_IRQ[i] @(0xc) ; sert à déclencher une interruption.

Question C2

Le composant **PibusIcu** est une cible car il n'a pas une « Capacité DMA ».
Arguments du constructeur :
- nriq : indique le nombre de lignes d'interruptions (actives) en entrée.
- nproc : indique le nombre de cœurs, donc de lignes d'interruptions en sortie de l'ICU.

L'OS peut aiguiller la ligne d'interruption connectée à l'entrée IRQ_IN[i] vers le processeur connecté à la sortie IRQ_OUT[j] en utilisant un registre 32 bits comme masque. IRQ_IN[i] est activée si IRQ_MASK[i] = 1.

Les registres associés à chaque port de sortie IRQ_OUT[i] :
- ICU_INT @(0x00); indique le numéro de l'interruption, lignes actives/inactives.
- ICU_MASK @(0x04); indique le masque associé à l'interruption en cours.
- ICU_MASK_SET @(0x08); activer, pour chaque cœurs ces lignes actives d'interruptions.
- ICU_IT_RESET @(0x0c); désactiver, pour chaque CPU, ces lignes d'interruptions.
- ICU_IT_VECTOR @(0x10);contient l'index de l'interruption active la plus prioritaire.

Question C3

La segment correspondant au PibusIcu a pour taille « 32 * nprocs » où nprocs peut aller jusqu'à 8. Alors qu'un segment soit bien aligné en mémoire, il faut bien que son adresse de base soit multiple de la taille de l'objet.
(?)Non-alignement des adresses cause des complications au niveau matériel depuis que la mémoire soit alignée sur la limite d'un mot. Donc, un objet mal-aligné prendra plusieurs références mémoire alignées.
(http://web.cs.iastate.edu/~prabhu/Tutorial/PIPELINE/addressAlign.html).

Question C4

Les 4 lignes d'interruptions sont connectées sur les ports du contrôleur ICU en indiquant les signaux en entrée :

```
icu.p_irq_in[0]             (signal_irq_dma);
icu.p_irq_in[1]             (signal_irq_ioc);
for ( size_t i=0 ; i<nprocs ; i++)
{
    icu.p_irq_in[2+2*i]     (signal_irq_tim[i]);
    icu.p_irq_in[3+2*i]     (signal_irq_tty_get[i]);
    icu.p_irq_out[i]        (signal_irq_proc[i]);
}
```

D) *Lancement des tâches*

Question D1

Le fichier « reset.s » ainsi modifié :

```
# initializes stack pointer for PROC[1]
la      $29,    seg_stack_base
li      $27,    0x20000
addu    $29,    $29,    $27

# initializes SR register for PROC[1]
li      $26,    0x0000FF13
mtc0    $26,    $12

# jump to main in user mode: main[1]
la      $26,    seg_data_base
lw      $26,    4($26)
mtc0    $26,    $14
eret
```

Question D2

- L'adresse de main_prime() : 0x4012dc
- L'adresse de main_pgcd() : 0x4013f0

Question D3

Pour forcer GCC à créer une un table de sauts au début du segment data, on utilise l'attribut « __attribute__((constructor)) ». En effet, le fichier partagé .o contient des sections (.ctors et .dtor dans ELF) qui possède des références sur les fonctions déclarées avec l'attribut contructor. Quand la librairie soit chargée, le loader va vérifier s'il telles sections existes, donc ce cas, il appelle les fonctions référencées via cette attribut.
https://www.geeksforgeeks.org/__attribute__constructor-__attribute__destructor-syntaxes-c/

Question D4

Le programme de calcul du PGCD reste bloqué sur la saisie de l'opérande X car les interruptions ne sont pas activées et que l'appel à la fonction TTY_GETW_IRQ() est bloquant.

E) *Activation du Timer*

Question E1

Lorsque le processeur reçoit un signal d'interruption, il écrit dans le registre cause le code correspondant au type de l'appel(dans notre cas, c'est une interruption externe dont le code est le « 0000 ») et se branche dans le GIET.
La séquence d'appels de fonction entre le branchement dans le GIET et le branchement à la routine isr_timer :
- Le GIET récupère le code de l'appel dans le registre cause, et via le tableau « _cause_vector » indexé par le ce registre, il détermine que c'est une interruption et se branche à l'adresse correspondant à l'étiquète « int_handler ».
- Le GIET sauvegarde tout les registres non persistants dans le pile du programme interrompu et se branche dans la fonction « _int_demux.c » pour trouver la bonne ISR. Au retour de cette fonction, il restaure les valeurs de ces registres.
- La fonction « _int_demux » récupère l'index de l'interruption la plus prioritaire en lisant le registre « ICU_IT_VECTOR » du composant ICU, et appelle la bonne ISR.
Le vecteur d'interruption est initialisé à l'adresse de la routine par défaut « isr_default ». Le code de boot se charge d'écrire l'adresse des interruptions actives dans ce vecteur.

Question E2

La routine « _isr_timer » traite au maximum 8 IRQ générées par 8 timers indépendant. Le comportement dépend du processeur interrompu.
Elle acquitte l'interruption du timer[proc_id] du processeur interrompu et affiche un message sur le bon TTY[proc_id].

Question E6

- Le processeur 0 écrit la première valeur « 0x80001bf8 (&_isr_timer) » dans le vecteur d'interruption « 0x82000324 (_interrupt_vector[3]) » au cycle 57.
- Le MASK[0] de l'ICU est configuré au cycle 82 (écriture de 4 dans 0x9f000008).
- Le Timer[0] est configuré au cycle 87 (écriture de 50000 dans 0x91000008).

Question E7

Le processeur reçoit la première interruption au cycle 50095 (TIM_IRQ[0] passe à 1).
Elle est acquittée par l'ISR au cycle 50602 (TIM_IRQ[0] repasse à 0).

Question E8

Le traitement d'une interruption Timer par le processeur 0 :
- Le signal PROC_IRQ[0] est activé au cycle 50094
- Le processeur se branche au point d'entrée du système d'exploitation GIET au cycle 50097
- Le processeur se branche au gestionnaire d'interruptions au cycle 50120
- Le processeur se branche à la fonction INT_DEMUX au cycle 50273
- Le processeur se branche à _ICU_READ au cycle 50322
- Le processeur revient à _INT_DEMUX au cycle 50463
- Le processeur se branche à l'ISR _ISR_TIMER au cycle 50521
- Le processeur revient à _INT_DEMUX au cycle 54627
- Le processeur restaure la valeurs des registres au cycle 54647
- Le programme interrompu est repris au cycle 54718

Le nombre de cycles total : 4624c cycles.

F) *Activation des interruptions TTY*

Question F2

La fonction qui permet de récupérer la valeur d'un nombre saisi au clavier est « TTY_GETW_IRQ » :
- On reste bloquer jusqu'à ce que un caractère soit saisi au clavier, la fonction « TTY_READ_IRQ » qui elle même appelle la fonction « _PROCID », test si un caractère est disponible, si le cas, le recopier dans le tampon du programme user et réinitialise « _TTY_GET_FULL ».
- La fonction fait des traitement sur le nombre saisi au clavier pour produire une chaine de caractères décimaux. « TTY_PUTC » permet d'afficher la valeur dans le TTY.

Question F3

Si le tampon « TTY_GET_BUF » est plein, la valeur stockée sera écrasée par la nouvelle valeur récupérée depuis TTY_READ.

Question F4

La fonction « TTY_READ_IRQ » teste si un caractère est disponible, et le recopier dans le tampon user tout en remettant « TTY_GET_FULL » à zéro. Elle à pour arguments, un buffer du programme user et sa taille. Si le buffer est vide, la fonction va retourner zéro. Le numéro du terminal TTY concerné est calculé en prenant l'id du processeur pour indexer le tableau « _TASK_CONTEXT_ARRAY ».

Question F5

Les caractères spéciaux analysés par cette appel système :
- LF(0x0A), CR(0x0D) et DEL(0x7F).
Si le nombre de caractères décimaux saisis au clavier défini un nombre trop grand, la fonction supprime la chaine construite et retourne 0.

Question F6

Ces variables doivent être déclarer avec l'attribut VOLATILE car on veut qu'elles soient stockées non pas dans les registres du processeurs, mais dans la mémoire.
Le mot clé VOLATILE permet d'éviter les optimisation faites par le compilateur.
Le segment est non cachable car il s'agit des variables correspondant aux registres adressables des périphériques.