Ordinateurs, structure et applications
GIF-1001, Hiver 2015

TRAVAIL PRATIQUE NO 5
Réponse à un appel de service

Ce travail pratique vaut 3% de la note totale du cours. À faire individuellement, il sera corrigé sur un total de 10 points. Le travail est à remettre au plus tard mercredi le 1er avril à 23h59. La procédure de retard détaillée dans le plan de cours s'applique.

OBJECTIFS

Ce travail pratique vise les objectifs suivants :

- Comprendre comment fonctionnent les interruptions d'un ordinateur
- Comprendre ce qu'est un appel de service du système d'exploitation

ÉNONCÉ

Ce travail pratique est divisé en deux étapes principales :

1. Écrire un court code en assembleur ARM relié à l'instruction SWI pour un cœur ARM7TDMI (comme pour les TPs précédents) ;
2. Décrire la gestion des interruptions par un cœur ARM-Cortex M3.

Étape 1, code assembleur

Votre code assembleur doit mettre le numéro d'un appel de service dans le registre R0, lorsqu'un appel de service se produit. Les appels de service sont générés par les applications de l'usager, lorsque ces applications utilisent l'instruction SWI #X. L'instruction SWI déclenche une interruption logicielle. Vous devez implémenter la routine traitant l'interruption pour que R0 soit mis égal à X (R0 = X) lorsque l'interruption se produit. De plus, vous devrez écrire la bonne entrée dans la table des vecteurs d'interruption du système pour que les interruptions soient traitées.

Main	Sous-Routine traitant l'interruption logicielle SWI
main: SWI #0x112233 SWI #0x445566 B main	MonISRpourIntLogicielle : ;Votre code ici pour mettre R0 = 0x112233 ou 0x445566 ;Votre code ici pour mettre R0 = 0x112233 ou 0x445566 ;Votre code ici pour mettre R0 = 0x112233 ou 0x445566 ;Doit retourner au main…

Étape 2, description de la gestion des interruptions par ARM-Cortex M3

Pour décrire comment un microprocesseur ARM-Cortex M3 gère les interruptions, rappelez-vous que lors d'une interruption, les microprocesseurs, de manière générale, effectuent les opérations suivantes :

- Termine le ou les instructions en cours
- Vérifie si l'interruption peut être traitée (priorité et masque)

- Sauvegarde le contexte en cours
- Obtient l'adresse de la routine servant à traiter l'interruption (ISR)
- Effectue un saut à l'adresse de l'ISR

Vous devez décrire la façon dont chacune de ces étapes sont exécutées dans le cœur ARM-Cortex M3. À titre d'exemple, vous retrouverez la réponse qu'il faudrait donner pour le cœur ARM7TDMI ci-dessous : c'est le cœur utilisé par défaut dans les travaux pratiques et c'est celui pour lequel vous devez écrire le code ci-dessus. *Indice* : le document « ARM7TDMI.pdf », disponible sur le site web du cours, a été utilisé pour décrire les étapes ci-bas… trouvez l'équivalent pour le ARM-Cortex M3!

THÉORIE

Qu'est-ce qu'un appel de service (SWI)?

L'instruction SWI ou SVC (même chose que SWI, mais pour les microprocesseurs plus récents) est utilisée pour permettre au programmeur d'appeler des fonctions du système d'exploitation.

Habituellement, le programmeur ne sait pas le nom des fonctions ou les adresses des fonctions du système d'exploitation pour exécuter des tâches du OS. Par exemple, le programmeur ne sait pas l'adresse de la fonction permettant de lancer un thread ou de celle permettant d'afficher quelque chose à l'écran. En fait, le programmeur ne veut pas savoir les adresses de toutes les fonctions du OS…

Une façon facile d'appeler des sous-routines sans connaître leurs adresses est d'utiliser une interruption : le système d'exploitation remplit la table des vecteurs d'interruption pour que les programmeurs appellent les fonctions voulues lorsqu'ils lancent des interruptions logicielles.

L'instruction SWI #X lance une interruption logicielle. Lors du traitement de cette interruption, le système d'exploitation récupère la valeur de X et accomplit une tâche en fonction de celle-ci.

Comment récupérer X de SWI #X?

Lors de l'interruption logicielle, le registre LR contient l'adresse de retour. Puisque l'instruction avant celle à l'adresse de retour est forcément SWI #X, il est possible de lire les octets constituant cette instruction. Il est donc possible d'extraire #X…

La valeur de X est sur 24bits, l'opcode de SWI (0xef) étant sur 8 bits. Pour extraire X de l'instruction il faut masquer l'opcode afin de conserver les 24 bits les moins significatifs. "Masquer" signifie utiliser l'instruction AND avec les bits que nous voulons conserver à « 1 ». Par exemple, si vous voulez garder les quatre bits les moins signification de R0, vous

faites AND R0, R0, #0x0F. Si R0 valait 0x12345678 avant l'exécution de AND R0, R0, #0x0F, R0 vaudra 0x00000008 après l'exécution.

Comment le cœur ARM7TMDI gère-t-il les interruptions?

Lors d'une interruption, les microprocesseurs, de manière générale, effectuent les opérations suivantes :

- Termine le ou les instructions en cours
- Vérifie si l'interruption peut être traitée (priorité et masque)
- Sauvegarde le contexte en cours
- Obtient l'adresse de la routine servant à traiter l'interruption (ISR)
- Effectue un saut à l'adresse de l'ISR

Toutefois, ces opérations ne sont pas effectuées de la même manière par tous les microprocesseurs. Pour le ARM7TDMI, les actions suivantes sont effectuées :

- Termine le ou les instructions en cours
    - Le ARM7TDMI a un pipeline de 3 étages : Fetch, Décode, Exécute. Seule l'instruction à l'étape Exécute est terminée lorsque l'interruption survient. Les autres instructions sont annulées.
    - Les instructions plus longues du cœur (LDM et STM) sont interrompues en plein milieu pour assurer un temps court d'entrée dans l'interruption. Des drapeaux spéciaux du cœur permettent de reprendre l'exécution de ces instructions après l'interruption.
- Vérifie si l'interruption peut être traitée (priorité et masque)
    - La priorité des interruptions du cœur ARM7TDMI est la suivante (du plus prioritaire au moins prioritaire): Reset, Data Abort, FIQ, IRQ, Prefetch Abort, Undefined instruction et SWI.
- Sauvegarde le contexte en cours
    - Le microprocesseur ARM7TDMI utilise des banques de registres pour sauvegarder le contexte du programme interrompu. Il y a un registre LR, un registre SP et registre de drapeaux (SPSR) par contexte possible (une banque de registres) pour faire le lien avec les adresses de retour et avoir une pile par contexte.
        - Lors d'une interruption SWI/SVC par exemple, le registre LR utilisé sera LR_svc. La valeur mise automatiquement dans ce registre par le microprocesseur sera l'adresse de retour de l'ISR lorsque l'interruption SWI/SWC sera terminée.
        - Les autres registres du contexte doivent être sauvegardées sur la pile de l'interruption et remis à leur valeur initiale à la fin de l'ISR.
- Obtient l'adresse de la routine servant à traiter l'interruption (ISR)

- La table des vecteurs d'interruption du ARM7TDMI contient 8 valeurs de 32 bits, chaque valeur étant l'instruction à exécuter lorsque l'interruption survient, plutôt que l'adresse de la routine de l'interruption à traiter (une vraie table de vecteurs d'interruption contient des adresses plutôt que des instructions!).
- La table des vecteurs d'interruption est à l'adresse 0 et contient les vecteurs suivants :

Adresse	« Vecteur » d'interruption
0x00000000	Reset
0x00000004	Undefined Instruction
0x00000008	SWI ou SVC
0x0000000C	Prefetch Abort
0x00000010	Data Abort
0x00000014	Reserved
0x00000018	IRQ
0x0000001C	FIQ

- Effectue un saut à l'adresse de l'ISR
    - Lors d'une interruption, PC est mis à l'adresse de la table des vecteurs d'interruption correspondant à la nature de l'interruption survenue. Par exemple, lors de Reset, PC sera mis à 0. Lors d'une IRQ (interruption matérielle), PC deviendra automatiquement 0x18.
        - Les instructions qu'on retrouve dans la table des vecteurs d'interruption sont habituellement des instructions de branchement (B), sauf pour le FIQ…

<u>AUTRES EXIGENCES</u>

Remettez le fichier **asm.s** contenant votre code, avec la description de la gestion des interruptions du Cortex-M3 en commentaire à la fin du fichier.

Tout programme sans commentaires se verra attribuer la note de 0. Il faut au moins un commentaire toutes les trois lignes d'assembleur.

Tous les registres doivent conserver leurs valeurs lors d'un appel de fonction ou de routine de traitement d'interruption, à moins que ceux-ci ne soient utilisés comme paramètres de sortie. Autrement dit, la valeur des registres doit être conservées dans la fonction (Registres = {X}; BL MaFonction; Registres = {X} encore).

Bon travail!