TP5 & 6

Entrées/sorties ; interruptions

1- Entreés/Sorties

Un processeur a besoin de dispositifs d'entrées/sorties pour communiquer avec l'extérieur.

« Craps » et son simulateur disposent d'entrées/sorties minimalistes :

- Entrées : 16 switches (interrupteurs permettant d'entrer 0 ou 1) : La lecture des 16 switches se fait via l'adresse 0x90000000 avec l'instruction ld.
- Sorties : 16 leds permettant d'afficher 16 bits en parallèle : L'écriture sur les leds se fait via l'adresse 0xB0000000 avec l'instruction st

Le code suivant permet de lire les switches en continu et d'afficher la valeur lue sur les leds :

```
SWITCHES = 0x90000000
LEDS = 0xB0000000
                set SWITCHES, %r19
                set LEDS, %r20
boucle:         ld [%r19], %r1
                st %r1, [%r20]
                ba boucle
```

Tester ce programme.

Ecrire et tester le sous-programme afficher_leds_7_0 qui affiche une valeur (8 bits), passée en paramètre dans %r1, sur les 8 leds de faible poids et met à 0 les 8 leds de fort poids.

Ecrire et tester le sous-programme afficher_leds_15_8 qui affiche une valeur (8 bits), passée en paramètre dans %r1, sur les 8 leds de fort poids et met à 0 les 8 leds de faible poids.

2- Interruptions

Le simulateur CRAPS dispose d'une seule ligne d'interruption appelée IT (bouton en haut à droite), dont le mécanisme de prise en compte est le suivant :

- l'occurrence d'une interruption provoque, après la terminaison de l'instruction courante, la mémorisation dans la pile de l'adresse de l'instruction qui suit (PC), et des indicateurs N, Z, V, et C
- un branchement à l'adresse 1 : À cette adresse, un sous-programme (handler) d'interruption doit être implanté, terminé par l'instruction reti.
- Après exécution du code du handler, reti a l'effet suivant :
  - Le sommet de la pile est dépilé dans les flags (N, Z, V et C)
  - Le sommet de la pile est dépilé dans PC

L'exécution de reti provoque donc le retour dans le programme interrompu, juste après l'instruction durant laquelle l'interruption a eu lieu.

Le code suivant, éclaté exprès sur deux colonnes, compte le nombre d'interruptions IT. A chaque appui sur le bouton IT, une variable « compteur_IT » est incrémenté et est affichée sur les leds.

```
LEDS = 0xB0000000
PILE = 0x200
                ba progp
handler_IT:     …
                …
                …
                reti

// forcément après le handler d'IT
progp:          set PILE, %sp
                set compteur_IT, %r1
                st %r0, [%r1]
pbcle:          nop
                nop
                nop
                ba pbcle
compteur_IT:    .word 0
```

```
                ba progp        // instruction à l'@ 0
handler_IT:     push %r1        // handler d'IT à l'@ 1
                push %r2
                push %r20
                set LEDS, %r20
                set compteur_IT, %r1
                ld [%r1], %r2
                inc %r2
                st %r2, [%r20]
                st %r2, [%r1]
                pop %r20
                pop %r2
                pop %r1
                reti    // retour dans le prog interrompu
```

Tester ce programme.

2- Commutation de tâches

Nous allons utiliser l'interruption IT pour implanter une version simplifiée de la commutation de tâches. C'est le travail que fait le système d'exploitation pour suspendre, à intervalle de temps régulier, le processus en cours et donner la main au processus élu.

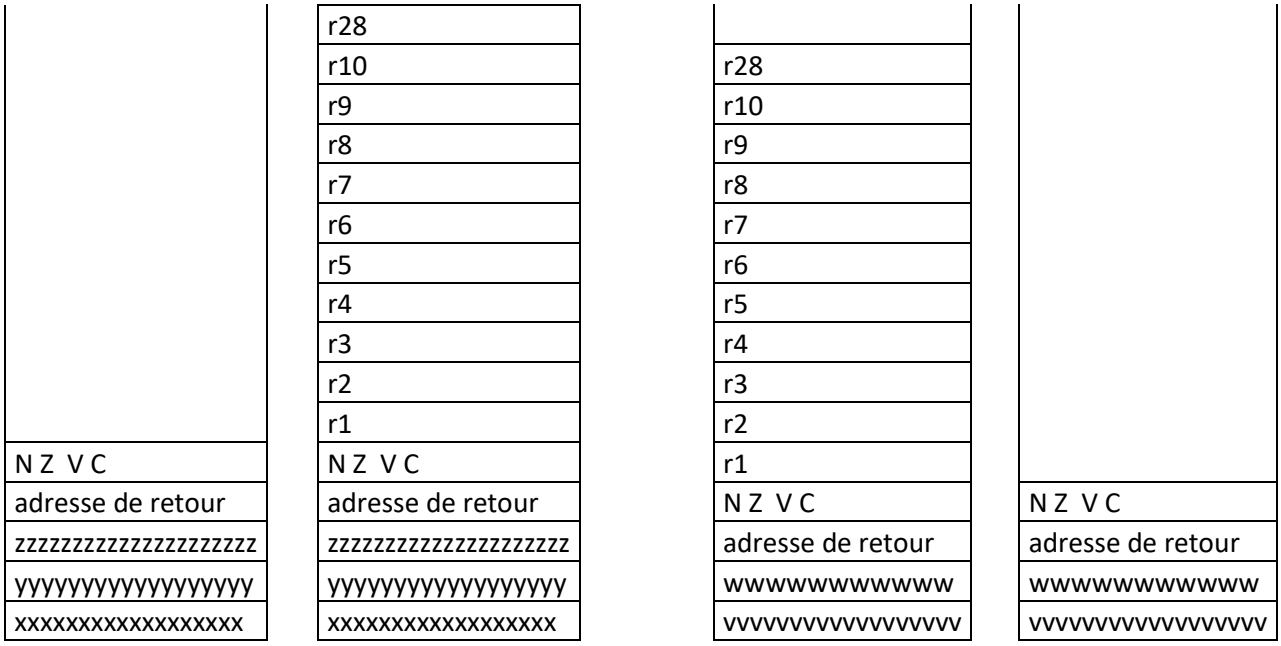
On suppose disposer de deux programmes susceptibles d'utiliser chacun tous les registres compris entre r1 et r10 (pour alléger le contexte sauvegardé).

Lorsque le programme1 est suspendu, tout son contexte doit être sauvegardé pour qu'il puisse être récupéré une fois il sera activé de nouveau, et reprendre son exécution comme si aucune interruption n'a eu lieu. Ce contexte est composé de tous les registres qu'il est supposé utiliser (on ne sait pas ce qu'il fait et ce qu'il utilise), y compris le registre %r28 (car il a le droit d'appeler des sous-programmes). Chaque programme dispose de sa propre pile.

On peut donc résumer l'algorithme du commutateur de tâches comme suit :

- Sauvegarder tous les registres de travail dans la pile du processus suspendu (qui est la pile courante)
- Sauvegarder le pointeur de pile dans un emplacement qui lui est associé
- Incrémenter le numéro du processus courant (stratégie simple)
- Récupérer le pointeur de pile du processus élu
- Récupérer tous les registres de travail du processus élu
- Reprendre l'exécution du processus élu

On peut voir ci-dessous l'évolution de la pile du processus suspendu et celle du processus élu, de l'entrée dans le commutateur jusqu'à la sortie.



On aura donc besoin de :

- Une variable qui contient le numéro du programme courant
- Une variable qui contient le nombre de programmes
- Un tableau de pointeurs de pile (Tab_sp) pour sauvegarder le pointeur de pile des processus après leur suspension

Pour simplifier l'implantation et les tests, on mettra ces informations dans des registres de façon permanente :

- R17 : contient le numéro du programme courant
- R18 : contiendra le nombre de programmes (ici 2)
- R19 : l'adresse Tab_sp

1- Ecrire le code du handler commutateur de tâches. Mettez en commentaire l'instruction qui incrémente le numéro du programme courant.
2- Ecrire le programme prog1, qui dans une boucle infinie :

- Incrémente un compteur et l'affiche sur les 8 leds de faible poids
- Réalise une temporisation d'environ 1 seconde (Ecrire et tester le sous-programme « wait » qui permet d'effectuer une attente de N secondes, N passée en paramètre)

3- Tester ce programme en cliquant un certain nombre de fois sur le bouton IT
4- Ecrire un second programme prog2 qui effectue le même travail que prog1, mais affiche son compteur sur les 8 leds de fort poids.
Décommenter l'instruction d'incrémentation, mise en commentaire plus haut, afin que le traitant de l'IT donne la main au prog2 lorsqu'on clique sur le bouton IT. Comment le traitant d'IT trouve-t-il ce contexte ? Ajouter ce qu'il faut pour initialiser ce contexte. Tester.

5- Généraliser avec 16 programmes, dont chacun fera clignoter une led qui lui sera associée.