Ordinateurs et Systèmes d'Exploitation
Exercices – X

1. À un instant donné, trois tâches d'un système d'exploitation sont chacune sur le point d'appeler un service différent du système : la première tâche appelle le service *exec*, la seconde le service *fork* et la troisième le service *exit*. En supposant que ces appels s'exécutent sans erreur, comment aura évolué le nombre de tâches du système après ces appels et pourquoi ?

   Solution

   Le nombre de tâches n'aura pas changé. En effet, l'appel du service *exec* ne crée pas de nouvelle tâche, seul le code exécuté par la tâche est modifié. L'appel du service *fork* crée bien une nouvelle tâche mais celui du service *exit* détruit la tâche appelante.

2. (a) Écrire un programme assembleur 80x86 effectuant, à l'aide de soustractions, la division entière d'un nombre naturel par un autre. La routine doit placer dans CX le résultat de la division de AX par BX et dans DX le reste de cette division. Les nombres sont positifs ou nuls et représentés sur 16 bits en représentation binaire non signée. On peut supposer que le contenu de BX est différent de 0 initialement. Seuls les registres CX, DX et FLAGS peuvent être modifiés par la routine.

   (b) En effectuant des appels à la routine ci-dessus, écrire un second programme assembleur 80x86 factorisant un nombre naturel donné dans le registre AX et supérieur ou égal à 2. Le registre BX pointe initialement vers un tableau de mots (16 bits) à remplir avec les représentations binaires non signées sur 16 bits des facteurs premiers de AX. Un facteur premier doit être répété dans le tableau s'il apparaît plusieurs fois dans la factorisation de AX. Le registre CX devra contenir en fin d'exécution le nombre de facteurs premiers, comptés avec leur multiplicité. Par exemple, si AX=12, le tableau devra contenir les valeurs 2, 2 et 3 (l'ordre n'a pas d'importance) et CX devra contenir la valeur 3. On peut supposer que le tableau est suffisamment grand pour contenir tous les facteurs.

   Solution

   (a) Après avoir mis CX à 0, on va soustraire BX de AX tant que BX reste inférieur à AX en incrémentant à chaque fois CX d'une unité. En sortant de cette boucle, CX contiendra le quotient et AX le reste. On utilise la pile pour sauver/restaurer les registres.

```
          SECTION .text
division: mov cx, 0
          push ax
boucle:   cmp ax, bx
          jb fin
          sub ax, bx
          inc cx
          jmp boucle
fin:      mov dx, ax
          pop ax
          ret
```

   (b) On utilise BX comme facteur potentiel de AX et on l'initialise à 2. Après appel de la routine de division, si le reste est nul, on ajoute BX à la liste des facteurs et on remplace AX par le résultat de la division. Sinon, on augmente BX d'une unité et on recommence le test de divisibilité.

```
            SECTION .text
            mov bp, bx
            mov bx, 2
            mov si, 0
boucle:     cmp ax, 1
            jne suite
            mov cx, si
            ret
suite:      call division
            cmp dx, 0
            je facteur
            inc bx
            jmp suite
facteur:    mov [bp], bx
            add bp, 2
            inc si
            mov ax, cx
            jmp boucle
```

3. Construire un circuit séquentiel capable de détecter un changement de valeur à son unique entrée i. Précisément, si t_1, t_2, t_3 dénotent les instants de trois coups d'horloge successifs quelconques, la valeur de l'unique sortie s doit être égale à 0 en t_3 si la valeur de l'entrée était identique en t_1 et t_2, et à 1 sinon.

Solution

On va utiliser un registre à deux bits, ceux-ci conservant la valeur de l'entrée lors des deux derniers coups d'horloge. La sortie sera alors simplement un XOR de la valeur courante des deux bits du registre.

Entrée	État courant		État suivant		Sortie
i	q_0	q_1	d_0	d_1	s
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	1	1	0

$$\begin{aligned} d_0 &= q_1 \\ d_1 &= i \\ s &= q_0 \oplus q_1 \end{aligned}$$