Systèmes Embarqués 1 & 2

Classes T-2/I-2 // 2018-2019

p.01 – Interruptions

Solutions

Exercice 1

Si le processeur se trouve en mode superviseur (svc), écrivez les instructions qui réalisent les opérations suivantes

(a) Passer du mode superviseur en mode système (sys)

(b) Placer la valeur 0x8020'0000 dans le pointeur de pile système (sys)

(c) Lire le long mot (32 bit) placé au sommet de la pile système

(d) Autoriser les interruptions IRQ, sans modifier les autres bits du registre d'état

(e) Passer du mode système au mode utilisateur (usr)

(f) Passer du mode utilisateur au mode superviseur

Solution:

```
// Passer du mode superviseur en mode système (sys)
    msr  cpsr_c, 0x9f

//Placer la valeur 0x8020'0000 dans le pointeur de pile système (sys)
    ldr  sp, =0x80200000

// Lire le long mot (32 bit) placé au sommet de la pile système
    ldr  r0, [sp]

// Autoriser les interruptions IRQ, sans modifier les autres bits du registre d'état
    mrs  r0, cpsr
    bic  r0, #0x80
    msr  cpsr_c, r0

// Passer du mode système au mode utilisateur (usr)
    mrs  r0, cpsr
    bic  r0, #0x1f
    orr  r0, #0x10
    msr  cpsr_cxsf, r0

// Passer du mode utilisateur au mode superviseur
    svc #1
```

Exercice 2

Si le µP se trouve en mode superviseur, écrivez le segment de code qui démarre un programme chargé à l'adresse 0x8000'1000. Le programme s'exécute en mode utilisateur. La pile utilisateur débute à l'adresse 0x8030'0000.

Solution:

```
msr  cpsr_c, 0xd0     // mettre en mode utilisateur
ldr  sp, =0x80300000  // charger le stack pointer
ldr  r0, =0x80001000  // charger l'adresse de depart
bx   r0               // appeler la routine
```

Exercice 3

Indiquez l'effet des instructions suivantes, en respectant l'ordre chronologique des instructions

```
msr   cpsr_cxsf, #0x93
eors  r0, r0
msr   cpsr_c, #0x93
msr   cpsr_c, #0x13
msr   spsr_cxsf, #0x93
movs  pc, lr
msr   cpsr_c, #0x91
msr   cpsr_c, #0x92
msr   cpsr_c, #0x90
msr   cpsr_c, #0x93
```

Solution:

```
msr cpsr_cxsf, #0x93    // --> CPSR=0x00000093
                        // charge le cpsr avec la valeur 0x93, flags=0, I=1, F=0, mode=SVC

eors r0, r0             // --> CPSR=0x40000093
                        // Z-flag=1

msr cpsr_c, #0x93       // --> CPSR=0x40000093
                        // change l'état des bits [7:0] I=1, F=0, mode=SVC

msr cpsr_c, #0x13       // --> CPSR=0x40000013
                        // I=0, F=0 IRQ & FIQ enabled

msr spsr_cxsf, #0x93    // --> SPSR=0x00000093
                        // charge le spsr avec la valeur 0x93

movs    pc, lr          // --> CPSR=0x00000093
                        // pc = lr, cpsr = spsr

msr cpsr_c, #0x91       // --> PSR=0x00000091
                        // change pour le mode FIQ

msr cpsr_c, #0x92       // --> CPSR=0x00000092
                        // change pour le mode IRQ

msr cpsr_c, #0x90       // --> CPSR=0x00000090
                        // change pour le mode utilisateur

msr cpsr_c, #0x93       // --> CPSR=0x00000090
                        // aucun effet, car pas possible de modifier en mode user
```

Exercice 4

Soit un mini système d'exploitation possédant deux jeux de 5 routines utilitaires. Ces routines sont accessibles par un programme utilisateur via l'instruction SVC #1 pour le 1er jeu et SVC #5 pour le 2e. Le programme utilisateur place dans le registre de donnée R0, le numéro d'identification (utility ID) avant l'appel au système d'exploitation. Ecrivez le segment de code qui implémente une telle approche du côté système d'exploitation et du côté utilisateur.

Solution: voir le code sous "exercices"

Exercice 5

Concevez un programme qui utilise les interruptions pour mesurer le temps qui s'écoule entre deux pressions de touche successives. La valeur minimale, la valeur maximale et le nombre de pressions de touche doivent être calculées.

Deux périphériques sont à disposition:

(a) Une horloge temps réel (real-time clock).
Ce périphérique émet une interruption à intervalle régulier

(b) Un clavier ou une souris.
Le périphérique utilisé émet une interruption à chaque pression d'une touche

La routine de service de l'horloge temps réel incrémente, à chaque interruption, une variable globale clock_count.

Solution:

```
// Une solution possible, en faisant abstraction de l'attachement des
// routines de traitement d'interruptions à leur vecteurs

uint32_t clock_count = 0;
uint32_t keyclick_count  = 0;
uint32_t last_clock_count   = 0;
uint32_t min_interval = -1;
uint32_t max_interval = 0;

void clock_isr()
{
  clock_count++;
}

void keyclick_isr()
{
    keyclick_count++;

    uint32_t clock_val = clock_count;
    uint32_t delta = clock_val - last_clock_count;
    if (keyclick_count > 1) {
      if (delta > max_interval) max_interval = delta;
      if (delta < min_interval) min_interval = delta;
    }
    last_clock_count = clock_val;
 }
```

Exercice 6

Quel est le résultat de l'instruction ci-dessous si l'interruption « irq_h » survient lors de son exécution ?

```
int len = 0;

len += 2;   // irq_h survient ici ?

void irq_h()
{
  len += 4;
}
```

Solution:

L'instruction len+=2; se décompose comme suit

```
        // <-- si irq arrive ici --> len == 6
ldr r0, =len
        // <-- si irq arrive ici --> len == 6
ldr r1, [r0]
        // <-- si irq arrive ici --> len == 2 !!!
add r1, r1, #2
        // <-- si irq arrive ici --> len == 2 !!!
str r1, [r0]
        // <-- si irq arrive ici --> len == 6
```

Exercice 7

Le µP est en train d'exécuter une application dans le mode utilisateur et les interruptions FIQ et IRQ sont autorisées. Que se passe-t-il et/ou quel est l'état du processeur, si

(a) une instruction SVC est appelée par l'application

(b) une interruption IRQ est levée puis une interruption FIQ est levée

(c) une interruption FIQ est levée puis une interruption IRQ est levée

Solution:

- une instruction SVC est appelée par l'application
  - Processeur entre dans le mode superviseur (SVC)
  - CPSR sauvé dans SPSR_svc, adresse de retour sauvée dans LR_svc
  - IRQ sont désactivées
- une interruption IRQ est levée puis une interruption FIQ est levée
  - Processeur entre dans le mode interrupt (IRQ)
  - CPSR sauvé dans SPSR_irq, adresse de retour + 4 sauvée dans LR_irq
  - IRQ sont désactivées
  - Processeur entre dans le mode fast interrupt (FIQ)
  - CPSR sauvé dans SPSR_fiq, adresse de retour + 4 sauvée dans LR_fiq
  - FIQ sont désactivées
- une interruption FIQ est levée puis une interruption IRQ est levée
  - Processeur entre dans le mode fast interrupt (FIQ)
  - CPSR sauvé dans SPSR_fiq, adresse de retour + 4 dans LR_fiq
  - FIQ et IRQ sont désactivées
  - IRQ n'a aucun effet, doit attendre la fin de traitement de la FIQ

Exercice 8

Expliquer la différence dans le traitement d'une interruption vectorisée et le traitement d'une interruption non-vectorisée. Quels sont les avantages et désavantages de ces deux types de traitement.

Solution:

Vectorisée

- Avantages
  - Simplicité logicielle (vecteur d'interruption en une lecture)
  - Performance (pas de temps de scrutation)
- Désavantages
  - Complexité matériel

Non-vectorisée

- Avantages
  - Simplicité matériel (peu onéreux)
- Désavantages
  - Complexité des algorithmes logiciels
  - Temps de scrutation élevé