Compte rendu

PFSI TD n°11 Exceptions

Abderahman KRIOUILE_G2

Objectifs : Comprendre et savoir utiliser les exceptions

1- Principes :

Les interruptions sont un mécanisme permettant à un circuit extérieur au microprocesseur d'interrompre le programme en cours d'exécution afin de faire exécuter une routine spécifique. Les interruptions sont en particulier utilisées pour la gestion des entrées/sorties et pour la gestion des processus.

2- Ecriture du programme principal :

Remarque : le programme fini est à la fin du compte rendu

```
//*****DIRECTIVES D'ASSEMBLAGE *****

SP          equ   R15          // SP est le  pointeur de la pile

MAIN_PRGA equ   0xFF10

INITIA      equ   0x0000       // valeur avec  laquelle on initialise cntm

MAX         equ   0x006F       // lorsque cntm dépasse cette valeur elle est
remis à zéro

PAS         equ   0x0010  // valeur ajoutée à cntm à chaque passage de la
boucle

CNTMA       equ   0xFF00  // adresse de la case mémoire dont le contenu
cntm est augmenté à chaque passage de la boucle

ORG         MAIN_PRGA      // chargement du programme dans la mémoire

START   debut       // adresse de la première instruction à exécuter
```

```
                //***** PROGRAMME PRINCIPAL*****
debut   LDW   SP, #BAS_PILE        // initialisation du pointeur de pile
        LDW   R0, #INITIA          // on initialise R0 à 0
        STW   R0, @CNTMA    // on initialise CNTMA à 0 en y écrivant R0
        LDW   R1, #MAX
LOOPA   ADQ   PAS, R0       / on ajoute l'incrément à R0
        CMP   R0, R1        // on compare R0 et R1, en calculant R0-R1)
        BLW   VAS-$-2       / si R0 <= MAX, on saute à la ligne VAS
        LDW   R0, #INITIA          // sinon, on réinitialise R0 à 0
VAS     STW   R0, @CNTMA    // on copie R0 à l'adresse CNTMA
        BMP   LOOPA-$-2            //enfin, on boucle
```

3- Ecriture du programme d'interruption :

A/

```
            CNTIA    equ  0xFF02   // adresse de la case mémoire dont le
contenu est incrémenté de 2 lors du sous-programme d'interruption
            Petit_PAS  equ  0x0002   // incrément
IT_PRGA STW   R0, -(SP)  //on empile R0, on utilise l'adressage basé pré-
décrémenté
            LDW   R0, @CNTIA        // on copie cnti dans R0
            ADQ   Petit_PAS, R0      // on ajoute Petit_PAS à R0
            STW   R0, @CNTIA        // on recopie R0 dans cnti
            LDW   R0, (SP)+          // on dépile R0
            ENI                     // on autorise à nouveau les interruptions
            RTI
```

B/ Ce sous-programme d'interruption utilise le registre R0. Il faut donc avant toute chose sauvegarder sa valeur dans la pile et la restituer à la fin du sous-programme.

C/ Il faut autoriser à nouveau les interruptions explicitement car elles ont été inhibées (IF à 0) implicitement dès le déclenchement du programme d'interruption.

D/ Cette machine efface IF (met IF à 0) lorsqu'elle lance le programme d'interruption pour ne pas interrompre les instructions urgentes du programme d'interruption en cours par d'autres éventuelles interruptions.

E/ RTS se contente de faire dépiler le contenu du PC de la pile, c'est l'instruction utilisé dans le cas d'un sous-programme normale mais dans le cas d'une interruption il faut en plus dépiler le contenu de SR on utilisera alors l'instruction RTI qui effectue les deux opérations à la fois.

4- Initialisations :

A/ le numéro INT d'interruption qui correspond à la ligne IRQ2 est : n = i + 32 = **34**

B/

```
//*****DIRECTIVES D'ASSEMBLAGE *****
SP          equ   R15              // SP est le  pointeur de la pile
STACKA      equ   0x1000 //initialisation de la pile à l'adresse STACKA=1000h
MAIN_PRGA equ    0xFF10
INITIA      equ   0x0000 // valeur avec  laquelle on initialise cntm
MAX         equ   0x006F // lorsque cntm dépasse cette valeur elle est remis à 0
PAS         equ   0x0010 // valeur ajoutée à cntm à chaque passage de la boucle
CNTMA       equ   0xFF00 // adresse de la case mémoire dont le contenu cntm est
```

```
                                    //augmenté à chaque passage de la boucle

VECTA  equ  0x0088 // adresse du vecteur d'interruption IRQ2 : 4 x (32+2) = 136d = 88h

ORG     MAIN_PRGA     // chargement du programme dans la mémoire

START   debut       // adresse de la première instruction à exécuter


                        //***** PROGRAMME PRINCIPAL*****

debut   LDW   SP, #BAS_PILE // initialisation du pointeur de pile

        LDW   R2,# IT_PRGA  // initialisation de la table des vecteurs d'interruption

        STW   R2, VECTA

        ENI                         //validation des interruptions

        LDW   R0, #INITIA           // on initialise R0 à 0

        STW   R0, @CNTMA    // on initialise CNTMA à 0 en y écrivant R0

        LDW   R1, #MAX

LOOPA ADQ   PAS, R0         // on ajoute l'incrément à R0

        CMP   R0, R1        // on compare R0 et R1, en calculant R0-R1)

        BLW   VAS-$-2       // si R0 <= MAX, on saute à la ligne VAS

        LDW   R0, #INITIA           // sinon, on réinitialise R0 à 0

VAS     STW   R0, @CNTMA            // on copie R0 à l'adresse CNTMA

        BMP   LOOPA-$-2             //enfin, on boucle
```

5- Sauvetage et restitution de l'état, instruction RTI :

A/ Entre «CMP R0, R1» et « BLW VAS-$-2 » car si un programme d'interruption est lancé SR changera et au retour l'instruction de saut BLW se basera sur des indicateurs faussés.

B/ RTI permet de restituer le registre d'état et la valeur du PC avant l'exécution de l'interruption. L'instruction RTI est spécialement conçue pour ce genre de situation contrairement au RTS classique.

6- Section critique, instruction DSI :

A/ Si CNTMA = CNTIA = FF00h, les deux programmes ne vont pas interagir comme prévu car si le programme principale s'interrompt entre le chargement et la réécriture de la valeur de la case mémoire, le résultat sera erroné.

B/ Pour résoudre ce problème il faut interdire les interruptions entre le chargement de la valeur de la case mémoire dans le registre et l'écriture de la nouvelle valeur dans la case mémoire après avoir effectué les calculs prévus. En utilisant l'instruction DSI qui met IF à 0.

```
debut   LDW   SP, #BAS_PILE
        LDW   R2,# IT_PRGA
        STW   R2, VECTA
        ENI
        LDW   R0, #INITIA
        STW   R0, @CNTMA
        LDW   R1, #MAX
LOOPA   DSI
        ADQ   PAS, R0
        CMP   R0, R1
        BLW   VAS-$-2
        LDW   R0, #INITIA
VAS     STW   R0, @CNTMA
        ENI
```

```
        BMP   LOOPA-$-2
```

7- Instruction HLT :

A/ HLT met le CPU en attente d' interruption, il s'arrête à l'instruction HLT jusqu'à la prochaine interruption.

B/ Une machine (CPU) en attente consomme moins de puissance électrique et donc chauffe moins.

8- Priorité et memorization :

A/

```
                NOV_PAS  equ  0x0001
NOV_PRGA        STW   R0, -(SP)
                LDW   R0, @CNTMA        // on copie cntm dans R0
                ADQ   -NOV_PAS, R0  // on soustraie NOV_PAS de R0
                STW   R0, @CNTMA        // on recopie R0 dans cntm
                LDW   R0, (SP)+
                ENI
                RTI
```

B/ En analysant la situation de plus prés c'est à dire au niveau de l'aspect matériel on voit que le programme d'interruption IRQ2 sera prioritaire car le bus IRQST sera liée à la masse simultanément à IRQ2 et IRQ3 et le périphérique le plus proche (IRQ2) block ITACK et attendra l'accusé de réception envoyé par le CPU pour envoyé son INT.

9- Trappe logicielle – Instruction TRP :

A/dans le programme principale on appelera le programme d'interruption du début en utilisant l'instruction suivante :

```
TRP    #34
```

B/ On utilise pas simplement l'instruction JSR car TRP n tient sur une seul instruction vu que les adresses sont exprimée par n qui est assez petite or JSR tient sur deux instructions car elle est suivi par l'adresse du sous programme qui s'étale sur 16 bit.

C/ L'intérêt de réaliser des programmes de trappe via un vecteur plutôt que des sous-programme est que ceci permet d'implémenter des appels système à numéro fixe défini par la documentation du système et qu'on peut appeler dans tout programme.

10- Trappe (matérielle) :

A/ Lorsqu'un programme effectue une action interdite le CPU déclenche une exception de type trappe dite matérielle.

B/ actions interdites susceptibles de déclencher une trappe :

- Tentative d'extraction de racine carré d'un nombre négatif.
- Tentative d'exécution de code d'instruction inexistante.
- Débordement de la pile.

11- Initialisation de l'unité central de traitement (CPU) :

A/ oui, l'initialisation est une exception.

B/ Elle est déclenché grâce au bouton ReSeT présent dans les machines et à l'instruction RST.

C/ Le programme en cours s'arrête sans retour et un programme d'initialisation est lancé depuis une adresse de démarrage fixe =FFFAh dans la machine RISK.

D/ non, l'initialisation n'est pas vectorisé dans cette machine.

E/ Le programme de démarrage de la machine est quelque part dans les adresses inférieurs de la mémoire c'est pourquoi on met une instruction de saut à l'adresse FFFA afin de lancer ce programme.

F/ L'instruction de saut utilise deux mots et puis 4 octets :

| FFFA | FFFB |

| FFFC | FFFD |

Ces 4 octets sont les derniers de la partie EPROM de la mémoire (mémoire non volatile).

Si on utilisait FFFE , le mot d'extension de l'instruction JPA tomberait en dehors de la mémoire.

G/ Si on utilisait une adresse plus faible, on sera obligé de prévoir des éventuels sauts dans nos programmes pour éviter ces octets d'initialisation sa sera mieux de les mettre à la fin de l'espace d'adressage pour ne pas déranger le programmeur, et dans le cas particulier d'un débordement imprévu la machine s'initialisera.

Voici le programme final comportant les deux programme d'interruption :

```
//*****DIRECTIVES D'ASSEMBLAGE *****
SP          equ   R15       // SP est le  pointeur de la pile
STACKA      equ   0x1000    //initialisation de la pile à l'adresse 1000h
```

```
MAIN_PRGA equ   0xFF10

INITIA      equ   0x0000   // valeur avec  laquelle on initialise cntm

MAX         equ   0x006F   // lorsque cntm dépasse cette valeur elle est remis à zéro

PAS         equ   0x0010   // valeur ajoutée à cntm à chaque passage de la boucle

CNTMA       equ   0xFF00   // adresse de la case mémoire dont le contenu cntm est augmenté à chaque passage de la boucle

VECTA1      equ   0x0088   // adresse du vecteur d'interruption IRQ2 : 4 x (32+2) = 136d = 88h

VECTA2      equ   0x008C   // adresse du vecteur d'interruption IRQ3 : 4 x (32+3) = 140d = 8Ch

CNTIA       equ  0xFF02   // adresse de la case mémoire dont le contenu est incrémenté de 2 lors du sous-programme d'interruption

 Petit_PAS   equ   2          // incrément du 1er sous programme

NOV_PAS     equ  -1          // incrément du 2ème sous programme

ORG     MAIN_PRGA     // chargement du programme dans la mémoire

START   MAIN_PRGA           // adresse de la première instruction à exécuter

              //***** PROGRAMME PRINCIPAL*****

debut   LDW   SP, # STACKA          // initialisation du pointeur de pile

        LDW   R2,# IT_PRGA          // initialisation de la table des vecteurs d'interruption

        STW   R2, @VECTA1

        ENI                         //validation des interruptions

        LDW   R0, #INITIA           // on initialise R0 à 0

        STW   R0, @CNTMA            // on initialise CNTMA à 0 en y écrivant R0
```

```
        LDW   R1, #MAX
LOOPA DSI
        ADQ   PAS, R0          // on ajoute l'incrément à R0
        CMP   R0, R1           // on compare R0 et R1, en
calculant R0-R1)
        BLW   VAS-$-2          // si R0 <= MAX, on saute à la ligne VAS
        LDW   R0, #INITIA      // sinon, on réinitialise R0 à 0
VAS     STW   R0, @CNTMA       // on copie R0 à l'adresse CNTMA
        ENI
        TRP   #34
        TRP   #35
        BMP   LOOPA-$-2        //enfin, on boucle


            //***** PROGRAMME D'INTERRUPTION n°1*****
IT_PRGA   STW   R0, -(SP)      //on empile R0, on utilise
l'adressage basé pré-décrémenté
            LDW   R4, @CNTIA   // on copie cnti dans R0
            ADQ   Petit_PAS, R4  // on ajoute Petit_PAS à R0
            STW   R4, @CNTIA   // on recopie R0 dans cnti
            LDW   R0, (SP)+    // on dépile R0
            ENI    // on autorise à nouveau les interruptions
            RTI
           //***** PROGRAMME D'INTERRUPTION n°2*****
NOV_PRGA      STW   R0, -(SP)
              LDW   R5, @CNTMA     // on copie cntm dans R0
              ADQ   NOV_PAS, R5    // on ajoute NOV_PAS à R0
```

```
                STW   R5, @CNTMA        // on recopie R0 dans cntm


                LDW   R0, (SP)+

                ENI

                RTI


                RSB  MAIN_PRGA -$

                JPA  #debut
```