.:: Architecture ::.

semaine n°7
Révisions – Partiel de 2004/2005

Partiel de 2005

- Exercice n°1 : questions diverses

Répondez en une phrase sous la question.

1. Quelle différence principale existe entre un microprocesseur et un microcontrôleur ?

2. Quel est le rôle de l'horloge dans un microprocesseur ?

3. Quelle est la différence entre un compteur et un temporisateur ?

4. Qu'est ce que l'adressage indirect par registre ; illustrez.

5. Que retourne une routine d'interruption ?

6. Combien de cycle faut-il pour exécuter l'instruction ANL R0,R1 ?

Partiel de 2005

- Exercice n°2 : Soit la fonction C suivante :

```
unsigned int filtre(long * tab, unsigned int n) {
// tab un tableau d'éléments de type long (32 bits)
// n est le nombre d'éléments du tableau
unsigned int i, nb=0;
for(i=0; i<n; i++)
   if(tab[i]==100) {
      tab[i]=0; nb++;
   }
return nb;
}
```

- Ecrivez cette fonction en assembleur : en respectant les conventions de passage d'arguments et de valeur de retour du compilateur MCS251 utilisé en TP.
- en n'utilisant que des instructions valides !!! en commentant chaque ligne, en cherchant à optimiser le code

Partiel de 2005

- Exercice n°2 : Soit la fonction C suivante :

```
•unsigned int filtre(long * tab, unsigned int n) {
•WR10                    WR0        WR2
•unsigned int i, nb=0;
•WR12              WR14             + tab[1] => DR16
•
•_filtre:    ;Sauve le registres utilisés par les var locales
•            PUSH  WR12; PUSH  WR14 ; PUSH WR16 ; PUSH WR18
•            XRL   WR14,WR14   ; nb = 0
•            XRL   WR12,WR12   ; i = 0
•wh1:        CMP   WR12,WR4    ; i < n ?
•            JGE   ewh1
•if1:        PUSH  WR12        ; calcule tab[i]
•            SLL   WR12        ; tableau de long
•            SLL   WR12
•            ADD   WR12,WR0
•            MOV   WR16,@WR12  ; chargement par mots de 16b
•            MOV   WR18,@WR12+2
•
•            CMP   DR16,#100; tab[i] == 100 ?
•            JNE   eif1
•
```

Partiel de 2005

```
th1:  MOV   WR16,#0              ; dans ce cas on remplace par 0
      MOV   @WR12,WR16           ; écriture par mots de 16b
      MOV   @WR12+2,WR16
      INC   WR14                 ; nb++
eif1: POP   WR12                 ; ne pas oublier de restituer i
      INC   WR12                 ; i++
      JMP   wh1
ewh1: MOV   WR10,WR14          ; pour retourner nb
      POP   WR18                   ; restitue les registres utilisés
      POP   WR16                   ; par les variables locales.
      POP   WR14
      POP   WR12
      RET
```

Partiel de 2005

•Nous devons réaliser un système d'alarme, son fonctionnement est simple, les entrées sont les suivantes : 4 interrupteurs (I0 à I3) sont reliés aux portes et fenêtres à surveiller, si l'une d'elles est ouverte, cette entrée retourne la valeur binaire 1 au système, sinon 0. Un cinquième interrupteur valide le système (V0), il retourne un 1 lorsque le système est activé, 0 sinon. Les sorties sont les suivantes : 2 indicateurs lumineux (S0 et S1).
•S0 est à 1 lorsque le système est actif, le second à 1 lorsqu'une porte ou fenêtre est ouverte alors que le système est actif.
•
•Ce système est réalisé à l'aide d'un microcontrôleur, les bits I0 à I3 peuvent être lus sur les 4 bits de poids faible de l'adresse 0xFFF900, le bit V0 est lu sur le bit de poids le plus fort de la même adresse.
•Les sorties S0 et S1 sont pilotées en écrivant sur les deux bits de poids faible de l'adresse 0xFFFA00. Les entrées et sorties fonctionnent en logique normale (positive).
•
•Ecrivez le programme assembleur qui réalise entièrement le système décrit.
•
•

Partiel de 2005

```
•
•_main : MOV    R4,#0                ; valeur à positionner sur les sorties
•        MOV    WR0,#0FFh            ; lire l'adresse FF:F900h
•        MOV    WR2,#0F900h
•        MOV    R0,@DR0              ; l'état des interrupteurs dans R0 V0 x x x I3 I2 I1 I0
•if1:    CMP    R0,#080h             ; si V0 = 0 ne pas tester les I3-I0
•        JL     eif1
•        ORL    R4,#1                ; positionne à un l'indicateur de validation du système
•        ANL    R0,#0Fh              ; clear des poids forts
•if2:        CMP    R0,#0                ; si R0 n'est pas nul alors au moins un inter est
ouvert
•        JE     eif2
•        ORL    R4,#2                ; positionne l'indicateur de fenêtre / porte ouverte
•eif1:
•eif2:   MOV    WR0,#0FFh            ; écriture à l'adresse FF:FA00
•        MOV    WR2,#0FA00h
•        MOV    @DR0,R4              ; le la valeur calculée des sorties
•        JMP _main                   ; while(1)
•
```

Partiel de 2005

-
- ***_Exercice n°4 : timer du MCS251***
-

Compléter le code de la fonction ci-dessous pour qu'elle lance une interruption du timer 1, 45 ms après la fin de son exécution.La partie du cours relative à cette question est jointe en annexe. Rappelons que le timer 1 utilisé en TP divise par 12 une fréquence de 8MHZ, soit une incrémentation toutes les 1,5 µs. Pour vous aider, répondez préalablement aux question suivantes :

Quel événement va déclencher l'interruption ?

Quel mode du timer 1 permet en un seul chargement de temporiser autant ?

Combien d'incrémentations sont nécessaires ?

-

```
void tempo45ms(void) {
  EA=1;          // n'interdit pas les interruptions
  ET1=1;          // autorise l'interruption du timer 1
•   ...
•}
```