.:: Architecture ::.

semaine n°4
Passage d'arguments

Interrogation

1 – Traduire en assembleur la fonction suivante

```
void updT1T2(char * tab1, int * tab2, int i, int j) {
    tab2[i] = tab1[j];
}
```

On considérera que les registres suivant sont deja initialisés :

l'adresse de tab1 est initialisée dans WR0
l'adresse de tab2 est initialisée dans WR2
i dans WR4
j dans WR6

Etapes :

1 – simplifier la fonction
2 – remplacer les variables par des registres
3 – traduire en assembleur

Interrogation

Traduire en assembleur la fonction suivante

```
void updT1T2(char * tab1, int * tab2, int i, int j) {
    tab2[i] = tab1[j];
}

{
    s = tab1[i]          R10 = @WR0 +  WR4
    s = (int) s          WR10 = R10
    tab2[j] = s          @WR2 + 2 * WR6 = WR10
}

    PUSH WR4
    ADD WR4,WR0
    MOV R10,@WR4
    POP WR4

   MOVS WR10,R10

    PUSH WR6
    ADD  WR6,WR6
    ADD WR6,WR2
    MOV @WR6, WR10
    POP WR6

    RET
```

Le MCS251-Passage d'arguments

- Voir cours ... P29 ... P30

Le MCS251-Passage d'arguments

- Synthèse du passage des arguments
  - Le programme C passe les arguments d'une fonction dans des registres et dans la pile.
  - L'affectation des registres par les arguments est faite dans le programme en C.
  - La fonction assembleur reçoit donc des registres déjà initialisés

Le MCS251-Passage d'arguments

- Synthèse du passage des arguments
  - La fonction en assembleur retourne son résultat en utilisant un registre.
  - Le programme assembleur devra donc affecter dans ce registre spécifique la valeur à retourner.

Le MCS251-Passage d'arguments

- Synthèse du passage des arguments
  - Une fonction qui modifie les registres 12 à 23 doit les restituer dans leur état initial. Les autres peuvent être modifiés
    - La fonction en assembleur doit donc les empiler avec PUSH en son début et les restituer avec POP à la fin.
  - Le compilateur utilise en priorité les registres R0 à R7 pour passer les arguments d'une fonction. Il utilise ensuite la pile.
  - Pour les variables locales de la fonction assembleur il faut favoriser les registres WR12 à WR22. (qui seront à sauvegarder/restituer)

Le MCS251-Passage d'arguments

- Synthèse du passage des arguments

- L'affectation des registres pour passer les arguments se fait suivant la règle du first fit
  - Le premier registre libre, de taille adaptée, est choisi parmi R0 à R7. La liste des arguments de la fonction étant parcourue de gauche à droite.
  - S'il n'y a pas de registre de taille adaptée disponible, les arguments sont passés par la pile.

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

```
int    fonction1 ( char * a1, int a2, char a3, int a4, char a5, int a6)

char fonction2 ( char * a1, int  * a2, long * a3, long * a4)

long fonction3 ( int a1, int a2, int a3, char a4 )

short fonction4 ( unsigned short a1, unsigned long a2, unsigned char a3 )

char * fonction5 ( char * a1, char * a2, int a3)

char fonction 6 ( long a1, long a2, char a3, char a4, char a5, char a6, int a7)
```

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

```
int    fonction1 ( char * a1, int a2, char a3, int a4, char a5, int a6)
WR10                  WR0   WR2      R4      WR6     R5      pile
char fonction2 ( char * a1, int  * a2, long * a3, long * a4)

long fonction3 ( int a1, int a2, int a3, char a4 )

short fonction4 ( unsigned short a1, unsigned long a2, unsigned char a3 )

char * fonction5 ( char * a1, char * a2, int a3)

char fonction 6 ( long a1, long a2, char a3, char a4, char a5, char a6, int a7)
```

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

```
int    fonction1 ( char * a1, int a2, char a3, int a4, char a5, int a6)
WR10                WR0   WR2      R4      WR6     R5      pile
char fonction2 ( char * a1, int  * a2, long * a3, long * a4)
R11                 WR0      WR2      WR4      WR6
long fonction3 ( int a1, int a2, int a3, char a4 )

short fonction4 ( unsigned short a1, unsigned long a2, unsigned char a3 )

char * fonction5 ( char * a1, char * a2, int a3)

char fonction 6 ( long a1, long a2, char a3, char a4, char a5, char a6, int a7)
```

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

int fonction1 (char * a1, int a2, char a3, int a4, char a5, int a6)
WR10 WR0 WR2 R4 WR6 R5 pile
char fonction2 (char * a1, int * a2, long * a3, long * a4)
R11 WR0 WR2 WR4 WR6
long fonction3 (int a1, int a2, int a3, char a4)
DR8 WR0 WR2 WR4 R6
short fonction4 (unsigned short a1, unsigned long a2, unsigned char a3)

char * fonction5 (char * a1, char * a2, int a3)

char fonction 6 (long a1, long a2, char a3, char a4, char a5, char a6, int a7)

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

int fonction1 (char * a1, int a2, char a3, int a4, char a5, int a6)
WR10 WR0 WR2 R4 WR6 R5 pile
char fonction2 (char * a1, int * a2, long * a3, long * a4)
R11 WR0 WR2 WR4 WR6
long fonction3 (int a1, int a2, int a3, char a4)
DR8 WR0 WR2 WR4 R6
short fonction4 (unsigned short a1, unsigned long a2, unsigned char a3)
WR10 WR0 DR4 R2
char * fonction5 (char * a1, char * a2, int a3)

char fonction 6 (long a1, long a2, char a3, char a4, char a5, char a6, int a7)

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

int fonction1 (char * a1, int a2, char a3, int a4, char a5, int a6)
WR10 WR0 WR2 R4 WR6 R5 pile
char fonction2 (char * a1, int * a2, long * a3, long * a4)
R11 WR0 WR2 WR4 WR6
long fonction3 (int a1, int a2, int a3, char a4)
DR8 WR0 WR2 WR4 R6
short fonction4 (unsigned short a1, unsigned long a2, unsigned char a3)
WR10 WR0 DR4 R2
char * fonction5 (char * a1, char * a2, int a3)
WR10 WR0 WR2 WR4
char fonction 6 (long a1, long a2, char a3, char a4, char a5, char a6, int a7)

Le MCS251-Passage d'arguments

Exercices :

Quels sont les registres utilisés pour les arguments des fonctions suivantes ?

```
int    fonction1 ( char * a1, int a2, char a3, int a4, char a5, int a6)
WR10                WR0   WR2      R4      WR6     R5      pile
char fonction2 ( char * a1, int  * a2, long * a3, long * a4)
R11                 WR0      WR2      WR4      WR6
long fonction3 ( int a1, int a2, int a3, char a4 )
DR8             WR0      WR2   WR4  R6
short fonction4 ( unsigned short a1, unsigned long a2, unsigned char a3 )
WR10                WR0                   DR4                 R2
char * fonction5 ( char * a1, char * a2, int a3)
WR10                WR0      WR2      WR4
char fonction 6 ( long a1, long a2, char a3, char a4, char a5, char a6, int a7)
R11                 DR0   DR4      pile      pile      pile      pile      pile
```

Le MCS251-Passage d'arguments

- Règles de transformation C --> Assembleur
  1) Ecrire l'algorithme en C
  2) Simplifier l'algorithme
  3) Identifier les registres utilisés pour le passage des arguments et la valeur de retour
  4) Affectez aux variables locales des registres, de préférence entre WR12 et WR22
  5) Traduisez en assembleur
     1) Empiler les registres WR12 à WR22 qui sont modifiés
     2) Traduire le corps de la fonction
     3) Affecter le résultat
     4) Dépiler les registres WR12 à WR22 précédemment sauvés
     5) Retourner

Exo

1 – Traduire en assembleur la fonction suivante

```
int abs(int * tab, int len) {
    int i, c=0;
    for ( i = 0 ; i < len ; i++ ) {
        if ( tab[i] < 0 ) {
            tab[i] = -tab[i];
            c++;
        }
    }
    return c;
}
```

Exo

1 – Traduire en assembleur la fonction suivante

```
int abs(int * tab, int len) {
    int i = 0, c = 0, w;
    while( i < len ) {
        int v = tab[i];
        if ( v < 0 ) {

            w = 0 – v;
            tab[i] = w;
            c++;
        }
        i++;
    }
    return c;
}
```

```
        WR0, WR2
        WR12 = 0, WR10=0
while ( WR12 < WR2 ) {
    WR14 = @WR0+2*WR12
    if ( WR14 < 0 ) {
        WR16 = 0
        WR16 = WR16 - WR14
        @WR0+2*WR12 = WR16
        WR10 ++
    }
    WR12++
}
ret
}
```

Exo

1 – Traduire en assembleur la fonction suivante

```
WR12 = 0                                PUSH WR12, PUSH WR14, PUSH WR16, MOV    WR12,#0
WR10 =0                                 MOV     WR10,#0
while ( WR12 < WR2 ) {          WH:     CMP     WR12,WR2
                                        JGE     EWH
  WR14 = @WR0+2*WR12                      PUSH    WR12
                                          ADD     WR12, WR12
                                          ADD     WR12,WR0
                                          MOV     WR14,@WR12
                                          POP     WR12
  if ( WR14 < 0 ) {             If1:      CMP     WR14,#0
                                          JSGE    Else1
    WR16 = 0                    Then1:      MOV   WR16,#0
   WR16 = WR16 – WR14                       SUB   WR16,WR14
   @WR0+2*WR12 = WR16                       PUSH  WR12
                                            ADD   WR12,WR12
                                            ADD   WR12,WR0
                                            MOV   @WR12,WR16
                                            POP   WR12
    WR10 ++                                 INC   WR10
  }                                         JMP   EndIf1
                                Else1:      NOP
  WR12++                        EndIf1:   INC   WR12
 }                                        JMP   WH
  ret                           EWH:    POP WR16, POP WR14, POP WR12, RET
}
```