- ***Sujet :*** fonction d'addition pondérée de deux vecteurs
- ***Durée :*** 1h
- ***Tous documents autorisés***

1. PREAMBULE

- L'exercice suivant consiste à traduire un programme C en assembleur C6600 dans une optique d'optimisation du temps d'exécution. L'exercice d'examen sera légèrement plus complexe. Si vous souhaitez d'autres exercices d'entraînement, je peux déjà vous garantir que pour l'examen, il s'agira de l'un des algorithmes de la ***DSPLIB*** de Texas Instruments. Les sources étant fournis par Texas, à vous d'y jeter un coup d'œil (***C:\ti\dsplib_c66x_<version>\packages\ti\dsplib\src***) après installation de la bibliothèque sur vos machines. Quant à savoir lequel ...

- Toutes les variables seront allouées statiquement (adresses statiques inchangées connues dès le lancement de l'applicatif), sauf les variables locales et paramètres de fonctions qui seront gérées par registres de travail CPU. Aucune utilisation de la pile système ne sera faite.

- Dans une optique de simplification d'écriture du programme assembleur, nous utiliserons des références symboliques pour la dénomination des pointeurs vers les régions statiques de données. De plus, rappelons que lorsque nous appelons une fonction assembleur depuis une fonction C, les paramètres sont respectivement passés par les registres A4, B4, A6, B6, A8, B8 … l'adresse de retour par B3 et une éventuelle valeur de retour par A4 en fin de procédure assembleur. Les constantes flottantes pourront également être nommée par une écriture naturelle héritée du langage C, par exemple 1.0 afin de représenter la valeur 1 au format flottant.

2. MAIN EN C CANONIQUE

- Appels de fonctions d'additions pondérées de vecteurs en C et en assembleur :

```
#define SIZE 8

/* inputs and ouput vectors */
const float x1[SIZE] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0};
const float x2[SIZE] = {8.0, 7.0, 6.0, 5.0, 4.0, 3.0, 2.0, 1.0};
float y[SIZE];

/*
* main entry point
*/
void main ( void )
{
   /* canonical c */
   w_vec_sp_cn(x1, x2, 3.0, y, SIZE);

   /* canonical c6600 assembly */
   w_vec_sp_asm_cn(x1, x2, 3.0, y, SIZE);

   while(1);
}
```

3. VECTOR WEIGHTED ADD C CANONIQUE

- Fonction d'addition pondérée de deux vecteurs en C canonique :

```
/*
 * vector weighted add, canonical C implementation
 */
void w_vec_sp_cn ( const float *x1,
                   const float *x2, const float m,
                   float *y,
                   const int nx)
{
   int i;

   for (i = 0; i < nx; i++)
      y[i] = (m * x1[i]) + x2[i];
}
```

4. MAIN ASSEMBLEUR C6600 CANONIQUE

- Appel de fonctions d'additions pondérées de vecteurs en assembleur C6600 (version commentée) :

```
SIZE    .macro
        8
        .endm

main :
            ; *** C :  w_vec_sp_cn(x1, x2, 3.0, y, SIZE);
            ; parameters
            MVKL        x1, A4
            MVKH        x1, A4
            MVKL        x2, B4
            MVKH        x2, B4
            MVKL        3.0, A6
            MVKH        3.0, A6
            MVKL        y, B6
            MVKH        y, B6
            MVKL        SIZE, A8
            ; return address
            MVKL        F1L1, B3
            MVKH        F1L1, B3
            ; function call
            B           w_vec_sp_cn
            NOP         5
F1L1 :
            ; *** C :  w_vec_sp_asm_cn(x1, x2, 3.0, y, SIZE);
            ; parameters
            MVKL        x1, A4
            MVKH        x1, A4
            MVKL        x2, B4
            MVKH        x2, B4
            MVKL        3.0, A6
            MVKH        3.0, A6
            MVKL        y, B6
            MVKH        y, B6
            MVKL        SIZE, A8
            ; return address
            MVKL        F1L2, B3
            MVKH        F1L2, B3
            ; function call
            B           w_vec_sp_asm_cn
            NOP         5
F1L2 :
            ; *** C :  while(1);
            B           F1L2
            NOP         5

            ; *** C :  }
            B           B3
            NOP         5
```

5. VECTOR WEIGHTED ADD ASSEMBLEUR C6600 CANONIQUE

- Fonction d'addition pondérée de deux vecteurs en assembleur C6600 (version commentée) :

```
; *** C :
; void w_vec_sp_asm_cn ( const float *x1,                  → A4
;                        const float *x2,                  → B4
;                        const float m,                    → A6
;                        float *y,                         → B6
;                        const int nx)                     → A8

w_vec_sp_asm_cn :
            ; *** C :  int i;
            MV          A8, A0
F2L1 :
            ; *** C :  for (i = 0; i < nx; i++)
            ;  ... x2[i];
            ;  ... x1[i];
            LDW         *A4++, A16
            NOP         4
            LDW         *B4++, B16
            NOP         4

            ; *** C :  … = (m * x1[i]) + x2[i];
            MPYSP       A16, A6, A16
            NOP         3
            ADDSP       A16, B16, B16
            NOP         3

            ; *** C :  y[i] = …
            STW         B16, *B6++
            NOP         4

            SUB         A0,1,A0
   [A0]     B           F2L1
            NOP         5
            ; *** C :  end for

            ; *** C :  }
            B           B3
            NOP         5
```

6. VECTOR WEIGHTED ADD ASSEMBLEUR VLIW C6600

- Fonction d'addition pondérée de deux vecteurs en assembleur C6600 avec optimisations assembleur propres aux architectures VLIW/EPIC. Avancement de branches d'exécution, utilisation d'instructions en parallèle, retirement des instructions NOP's :

```
; *** C :
; void w_vec_sp_asm_vliw ( const float *x1,          → A4
;                          const float *x2,          → B4
;                          const float m,            → A6
;                          float *y,                 → B6
;                          const int nx)             → A8

w_vec_sp_asm_vliw :
            ; *** C :  int i;
            MV          A8, A0
F2L1
            ; *** C :  for (i = 0; i < nx; i++)
            ;  ... x2[i];
            ;  ... x1[i];
 ||         LDW         *A4++, A16
            LDW         *B4++, B16
            NOP         4
            MPYSP       A16, A6, A16
            NOP         3
            ADDSP       A16, B16, B16
            NOP
            SUB         A0,1,A0
   [A0]     B           F2L1
            STW         B16, *B6++
            NOP         4
            ; *** C :  end for

            ; *** C :  }
            B           B3
            NOP         5
```

6. VECTOR WEIGHTED ADD ASSEMBLEUR VECTORIEL C6600

- Fonction d'addition pondérée de deux vecteurs en C avec déroulement de boucle d'un facteur 2 :

```
/*
 * vector weighted add, radix 2 C implementation
 * nx is a multiple of 2 and greater than or equal to 2.
 * x1, x2 and y are double-word aligned.
 */
void w_vec_sp_r2 ( const float *x1,
                   const float *x2, const float m,
                   float *y,
                   const int nx)
{
   int i ;
   float x11, x12, x21, x22, y1, y2;

   for (i = 0; i < nx; i+=2) {
      x11 = x1[i] ;
      x12 = x1[i+1] ;

      x21 = x2[i] ;
      x22 = x2[i+1] ;

      y1 = (m * x11) + x21;
      y2 = (m * x12) + x22;

      y[i] = y1;
      y[i+1] = y2;
   }
}
```

- Fonction d'addition pondérée de deux vecteurs en assembleur C6600 avec utilisation d'instructions vectorielles (DMPYSP, DADDSP, LDDW, STDW ...). Cette partie a été vue en travaux pratiques, je vous laisse donc la réaliser par vous même :

```
; *** C :
; void w_vec_sp_asm_r2 ( const float *x1,          → A4
;                        const float *x2,          → B4
;                        const float m,            → A6
;                        float *y,                 → B6
;                        const int nx)             → A8

w_vec_sp_asm_r2 :

            ; à vous de l'écrire !

            B           B3
            NOP         5
```