Travaux Dirigés n°4

Architecture des ordinateurs

—IR 2000—

Instructions et adressages

Cette séance de travaux dirigés est consacrée à l'étude d'exemples de programmes assembleurs. On travaillera sur le processeur virtuel DLX décrit dans JOHN.-L. HENNESSY et DAVID.-A. PATTERSON, *Architecture des Ordinateurs, une approche quantitative*, chez International Thomson Publishing, Paris, 1996 (voir feuille annexe pour le jeu d'instructions).

▸ **Exercice 1. (Instructions simples)**

Traduire en assembleur DLX le programme suivant :

```
int i, j, k;
i = 0; i = 1; i = 0xffffffff; i +=1; i = j + k; i = 2 * i + k * (j + 1);
```

1. En supposant que les variables `i`, `j`, `k` sont stockées dans des registres ;
2. En supposant que les variables `i`, `j`, `k` sont stockées en mémoire ;
3. Traduire en assembleur DLX le programme suivant :
   ```
   int i, j, *px, *py, t[10];
   i = j; i = *px; *px = i; px = py; *px=*py;
   t[i] = 2; t[2*i+j] = 2;
   px=t; (*px)++; px++; (*px)++;
   ```
4. Que pensez-vous du programme suivant :
   ```
   register int i, *px;
   *px = &i; (*px)++;
   ```
   Que faut-il supposer sur `i` ?

▸ **Exercice 2. (Comptage du nombre de bits)** Dans de nombreux microprocesseurs actuels, il n'y a pas d'instructions pour compter le nombre de bits à 1 (distance de Hamming) d'une suite de 2^n bits.

1. Écrire un programme C qui simule cette instruction à l'aide d'une boucle.

En fait, il est possible de la faire beaucoup plus rapidement, à l'aide de masques et d'additions :

2. Que se passe-t-il si l'on exécute l'instruction C suivante :
   ```
   R = (x & 0x5555) + (x & 0xAAAA) >> 1
   ```
3. En déduire un programme C qui compte le nombre de bits à 1 d'une variable x.
4. Traduire ce programme en assembleur DLX.

▸ **Exercice 3. (Boucles)**

1. Traduire en assembleur DLX le programme suivant :

```
int i, s, n;
n = 25;
for(i=0; i<n; i++) s=s + n;
```

2. Écrire en C puis en assembleur DLX la fonctions `strcpy` ; pour simplifier on suppose que `dest` et `src` sont dans `R1` et `R2`. Le résultat sera stocké dans R3.

```
char *strcpy(char *dest, const char *src);
```

3. Traduire en assembleur DLX la boucle du programme précédent (on suppose que x est stocké dans le registre `R1`).
4. Traduire en assembleur le programme suivant (les `int` sont stockés sur 32 bits) les variables `A`, `B` et `C` sont des variables globales :

```
#define N 1000
int A[N], B[N], C[N];
for(int i=0; i<N; i++) A[i] = B[i] + C[i];
```

▸ **Exercice 4. (Procédure simple, code en ligne)**

On veut compiler la fonction suivante :

```
void swap(int *a, int *b)
{
   int tmp;
   tmp = *a; *a=*b; *b=tmp;
}
void main(void)
{
   int i, j;
   swap(&i, &j);
}
```

Écrire les codes DLX en supposant que la fonction est codée

1. en ligne (`inline`) ;
2. en sous-programme, le passage des paramètres se faisant par registre ;
3. en sous-programme, le passage des paramètres se faisant dans la pile. On supposera que le pointeur de pile est dans le registre `R15`. On rappelle de plus que, en C, c'est le programme appelant qui se charge de dépiler les paramètres. Ceci permet de passer un nombre variable de paramètre.