

TD08 - GÉNÉRATION DE CODE X86

Dans chacun des exercices suivants, un court programme en L est donné ainsi que le code trois adresses lui correspondant. L'objet des exercices est d'écrire le code en assembleur x86 issu de la traduction du code trois adresses. On indiquera à tout moment de la génération, l'information contenue dans chacun des quatre registres **eax**, **ebx**, **ecx** et **edx**. Pour cela, on présentera la solution sous la forme d'un tableau contenant pour chaque instruction en code trois adresses l'instruction ou les instructions en assembleur lui correspondant. Ce tableau contient aussi quatre colonnes représentant les quatre registres, comme dans l'exemple suivant :

C3A	a	b	c	d	X86
t7 = 45 + 12	t7				mov eax, 45 add eax, 12
t8 = 3 * t7	t7	t8			mov ebx, 3 imul ebx, eax
...					...

1. APPELS DE FONCTION

Code-source L :

```

1 fct(entier a){
2   retour(a);
3 }
4 main ()
5 entier x;
6 {
7   x = fct(5);
8 }
```

Code trois adresses :

```

1 fct  fbegin
2      ret a
3      fend
4 main fbegin
5      param 5
6      t0 = call fct
7      x = t0
8      fend
```

2. EXPRESSION ARITHMÉTIQUE

Code-source L :

```

1 entier result;
2 main()
3 {
4   result = 1 * (2 + 3) -
5           (4 * (5 + 6));
6   ecrire(result);
7 }
```

Code trois adresses :

```

1 main  fbegin
2      t0 = 2 + 3
3      t1 = 1 * t0
4      t2 = 5 + 6
5      t3 = 4 * t2
6      t4 = t1 - t3
7      result = t4
8      write result
9      fend
```

3. BOUCLE ET OPÉRATIONS LOGIQUES

Code-source L : <pre> 1 entier n; 2 main() { 3 tantque n < 100 & 4 !(n < 0) faire { 5 n = lire(); 6 } 7 }</pre>	Code trois adresses : <pre> 1 main fbegin 2 e0 t0 = 1 3 if n < 100 goto e2 4 t0 = 0 5 e2 t1 = 1 6 if n < 0 goto e3 7 t1 = 0 8 e3 t2 = 1 9 if t1 == 0 goto e4 10 t2 = 0</pre>	<pre> 11 e4 if t0 == 0 goto e5 12 if t2 == 0 goto e5 13 t3 = 1 14 goto e6 15 e5 t3 = 0 16 e6 if t3 == 0 goto e1 17 t4 = read 18 n = t4 19 goto e0 20 e1 fend</pre>
--	---	--

4. TABLEAUX

Code-source L : <pre> 1 entier tab[10]; 2 main() 3 entier i; 4 { 5 i = 1; 6 tab[0] = 1; 7 tantque i < 10 faire { 8 tab[i] = tab[i - 1] + 1; 9 i = i + 1; 10 } 11 }</pre>	Code trois adresses : <pre> 1 main fbegin 2 i = 1 3 tab[0] = 1 4 e0 t0 = -1 5 if i < 10 goto e2 6 t0 = 0 7 e2 if t0 == 0 goto e1 8 t1 = i - 1 9 t2 = tab[t1] + 1 10 t3 = i 11 tab[t3] = t2 12 t4 = i + 1 13 i = t4 14 goto e0 15 e1 fend</pre>
---	---

5. RAPPEL : STRUCTURE DE LA TRAME DE PILE

$ebp + 8 + 4 * 3 - 0$	$ebp + 20$	paramètre 1
$ebp + 8 + 4 * 3 - 4$	$ebp + 16$	paramètre 2
$ebp + 8 + 4 * 3 - 8$	$ebp + 12$	paramètre 3
$ebp + 8$	$ebp + 8$	valeur de retour
$ebp + 4$	$ebp + 4$	instruction à effectuer après l'appel
		ancienne valeur de eip
	ebp	ancienne valeur de ebp
$ebp - 4 - 0$	$ebp - 4$	variable locale 1
$ebp - 4 - 4$	$ebp - 8$	variable locale 2
$ebp - 4 - 8$	$ebp - 12$	variable locale 3

- Une variable locale a : $ebp - 4 - a.adresse$
- Un argument a : $ebp + 8 + 4 * nb_args - a.adresse$
- La valeur de retour : $ebp + 8$