TD 7 Compilation

L3 INFO, Univ Lumière Lyon 2

2022 – 2023

Les exercices marqués avec (▶) sont notés, vous devrez les rendre pour évaluation. Votre rendu sera sous la forme d'un fichier de texte (vous pouvez utiliser MS Word ou Libreoffice) que vous déposerez dans l'espace moodle du cours. Les documents du cours sont autorisés. Les communications entre étudiants sont interdites.

▶ **Exercice 1** On considère le code incomplet de fonction suivante

```
long test(long x, long y, long z) {
    long val = __________ ;
    if ( __________ ){
        if( __________ )
            val = __________ ;
        else
            val = __________ ;
    } elseif ( __________ )
        val = __________ ;
    return val;
}
```

Le compilateur `gcc` produit le code assembleur qui suit (avec option `-Og`, uniquement les extraits relevants sont affichés)

```
test:
        movq    %rdx, %rcx
        leaq    (%rdx,%rsi), %rax
        subq    %rdi, %rax
        cmpq    $5, %rdx
        jle     .L2
        cmpq    $2, %rsi
        jle     .L3
        movq    %rdi, %rax
        cqto
        idivq   %rcx
        ret
.L3:
        movq    %rdi, %rax
        cqto
        idivq   %rsi
        ret
.L2:
        cmpq    $2, %rax
        jg      .L1
        movq    %rdx, %rax
        cqto
        idivq   %rsi
.
```

où les variables x, y et z sont dans `%rdi`, `%rsi` et `%rdx` respectivement.

Rappellez vous que l'instruction `lea` (load effective adresses) permet de deplacer des données entre opérands (vous pouvez la penser comme une version flexible de `mov` où on peut réaliser de calculs à la volée).

1. Que fait le code assembleur ?
2. Quel registre garde la valeur de la variable `val` ? Comment le savez-vous ?
3. Complétez le code de la fonction `test`. Copiez la fonction dans votre fichier de rendu.
4. Ajouter une fonction `main` au code obtenu afin de pouvoir effectuer un test de votre code. Copiez la fonction dans votre fichier de rendu. Prenez une capture d'écran montrant que le code compile correctement sur votre machine et produit le bon résultat.

▶ **Exercice 2** On considère le code incomplet de fonction suivante

```
long loop_while(long a, long b) {
    long result = _________ ;
    while( _________ ) {
        result _________ ;
        a = _________ ;
    }
    return result ;
}
```

Le compilateur `gcc` produit le code assembleur qui suit (avec option `-Og`, uniquement les extraits relevants sont affichés)

```
.LFB0:
        movl    $0, %eax
        jmp     .L2
.L3:
        leaq    (%rdi,%rsi), %rdx
        addq    %rdx, %rax
        subq    $1, %rdi
.L2:
        cmpq    %rsi, %rdi
        jg      .L3
        ret
```

où les variables a, et b sont dans `%rdi` et `%rsi` respectivement.

1. Que fait le code assembleur ?
2. Quel registre garde la valeur de la variable `val` ? Comment le savez-vous ?
3. Complétez le code de la fonction `test`. Copiez la fonction dans votre fichier de rendu.
4. Ajouter au code obtenu une fonction `main` afin de pouvoir effectuer un test de votre code. Copiez la fonction dans votre fichier de rendu. Prenez une capture d'écran montrant que le code compile correctement sur votre machine et produit le bon résultat.

Exercice 3 (*Analyse lexicale*)
Considérez l'ensemble de règles ci-dessous :

```
%{
%}
integer      [0-9]+
real         [0-9]+\.[0-9]*|\.[0-9]+
id           [a-zA-Z_][0-9a-zA-Z_]*
%%
{real}       { fprintf(stderr,"REAL [%s]\n",yytext); }
{integer}    { fprintf(stderr,"INTEGER [%s]\n",yytext); }
{id}         { fprintf(stderr,"ID [%s]\n",yytext); }
\n           { fprintf(stderr,"NEW_LINE [%s]\n",yytext); }
.            { fprintf(stderr,"UNKNOWN [%s]\n",yytext); } ]
```

1. Écrivez un parseur avec flex qui contient ces règles. Prenez une capture d'écran montrant que votre parseur est bien (vous pouvez faire `ls` pour montrer qu'il est présent dans votre machine).
2. Créez un deuxième fichier avec le contenu suivant

   ```
   mavariable=15/2.4;
   _resultat+=mavariable
   ```

   Prenez une capture d'écran affichant le contenu de votre fichier avec la commande `cat`.
3. Utilisez le parseur du point 1 pour analyser le contenu du fichier produit dans le point 2 (prenez une capture d'écran montrant le résultat de l'analyse).
4. Expliquer le résultat de l'analyse.