**SCR.2.1 TP 16 ⊥ :**

**Programmation en langage d'assemblage pour l'architecture ARMv8**

**Structure générale d'un programme – Invocation des appels système**

1. Le système d'exploitation est 32 ou 64 bits ?
2. Quelle est l'architecture de la machine ? Quel est le modèle du processeur ?

L'exemple simple suivant a pour objectif de montrer la structure générale d'un programme écrit pour l'assembleur, ainsi que le mécanisme qui permet au programme de communiquer avec le système d'exploitation (appels système), pour la réalisation d'un affichage à l'écran, par exemple.

On utilisera `gdb` pour analyser les programmes pendant qu'ils tournent : le contenu des registres, de la mémoire, etc.

```
#include<unistd.h>
#include<stdlib.h>

int main(void){

  write(1,"Hello World!\n",13);
  exit(0);
}
```

1. On saisit le texte du programme, disons dans le fichier "`printHW.s`". Voici le programme en langage d'assemblage. On lit les commentaires insérés pour comprendre ce qu'on y fait.

```
/*
    Lines enclosed here are comments

    A small program: prints "Hello World!"
*/

// From // to the end of this line is a comment: next are directives to the assembler

   .equ SYS_EXIT, 93
   .equ SYS_WRITE, 64

   .data // tells assembler to assemble the following in the data section
msg:  .asciz "Hello World!\n" //msg retains the address of the string

   .text      // tells assembler to assemble the following
              // in the text (code) section

   .globl _start // _start is there where the program starts,
                 // .globl makes it visible to the linker

_start:
  mov x0,#1     // value 1 is placed in register x0
  adr x1,msg    // the address retained by label msg is placed in register x1
  mov x2,#13
  mov w8, #SYS_WRITE   // svc must find the syscall number in w8
                       // and the syscall arguments in x0,x1,x2
  svc #0         // invoke syscall: displays on the screen
```

```
mov x0, #0     // in x0 put the value you want to exit with
mov w8,#SYS_EXIT
svc #0

.end
```

La démarche à faire pour réaliser les appels système :

— Repérer le numéro de l'appel système dans le fichier `/usr/include/asm-generic/unistd.h`.

— Consulter la page du manuel de l'appel système pour voir comment il utilise les arguments.

2. On assemble, on "link" :

```
$ as -gstabs -o printHW.o printHW.s
```

Si l'assembleur n'est pas d'accord, corriger ce qui ne va pas.

```
$ ld -O0 printHW.o
```

On obtient un exécutable sous le nom `a.out` (utiliser l'option `-o` de `ld` si on veut un autre nom pour son exécutable).
À l'aide de la commande `file`, vérifier le type du fichier exécutable construit.

3. Tester en lançant l'exécutable.
4. Analyser son programme pendant qu'il s'exécute :

```
$ gdb a.out
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
...
...
This GDB was configured as "aarch64-linux-gnu".
...
...
Reading symbols from a.out...
(gdb) run
Starting program: ../../..../a.out
Hello World!
[Inferior 1 (process 543510) exited normally]
(gdb) l 10,15
10
11 _start:
12         mov x0,#1
13         adr x1,msg
14         mov x2,#13
15         mov w8,#SYS_WRITE
(gdb) b 12
Breakpoint 1 at 0x4000b0: file printHW.s, line 12.
(gdb) run
Starting program:  ../../..../a.out

Breakpoint 1, _start () at printHW.s:12
12         mov x0,#1
(gdb) i r x0 x1 x2 w8
x0             0x0                0
x1             0x0                0
x2             0x0                0
w8             0x0                0
(gdb) step
13         adr x1,msg
```

```
(gdb) i r x0 x1 x2 w8
x0             0x1                 1
x1             0x0                 0
x2             0x0                 0
w8             0x0                 0
(gdb) s
14         mov x2,#13
(gdb) i r x0 x1 x2 w8
x0             0x1                 1
x1             0x4100d0            4260048
x2             0x0                 0
w8             0x0                 0
(gdb) x /s 0x4100d0
0x4100d0: "Hello World!\n"
(gdb) x /14cb 0x4100d0
0x4100d0: 72 'H'101 'e'108 'l'108 'l'111 'o'32 ' '87 'W'111 'o'
0x4100d8: 114 'r'108 'l'100 'd'33 '!'10 '\n'0 '\000'
(gdb) s
15         mov w8,#SYS_WRITE
(gdb) i r x0 x1 x2 w8
x0             0x1                 1
x1             0x4100d0            4260048
x2             0xd                 13
w8             0x0                 0
(gdb) s
16         svc #0
(gdb) i r x0 x1 x2 w8
x0             0x1                 1
x1             0x4100d0            4260048
x2             0xd                 13
w8             0x40                64
(gdb) s
Hello World!
18         mov x0,#0
(gdb) c
Continuing.
[Inferior 1 (process 543556) exited normally]
(gdb)
```