COMPUTER ARCHITECTURE
2020 - 2021

TD n°6

Exercise 1:

Let the following assembly code be used:

```
ADD   Im R1  -10
ADD   Im R1  100
JMPZ ET1
JMPN ET2
PUSH Rg1 R1
STOP
ET1:   STORE D R1 100
STOP
ET2:  STORE D R1 50
STOP
```

Where Im indicates that the addressing mode used is.

Signed numbers are represented in complement to 2 on 8 bits.

1. At the beginning of the program, R1 contains 20. After execution of the latter, where is R1 written? Why?
2. At the beginning of the program, R1 contains -90. After executing the program, where do you write R1? Why?
3. At the beginning of the program, R1 contains -120. After executing the program, where do you write R1? Why?

Exercise 2:

You write a program that can be either compiled or interpreted.

The compiler can compile ten thousand lines of code per second, then it takes two seconds to edit the links. Your final executable program contains twenty times more processor instruction lines than the advanced language start program and runs at a speed of one hundred thousand instructions per second.

To execute it, you also have the choice of using an interpreter working at a speed of one hundred thousand instructions per second. Unfortunately, the interpretation is complicated and requires two hundred times more instructions than your initial program.

1. Once the program is written (a thousand lines of code), how long does it take to run if it is compiled (including compilation time)? What if it is interpreted?
2. What are the two execution times if it contains one million lines of code?
3. When to choose compilation over interpretation, or vice versa?

Exercise 3:

A processor has registers *r0* to *r9*. We want to calculate the expression:

$$((r1 \times r2 - r3) / (r1 + r4 + r5)) + r6 + r2$$

and put the result in *r0*.

1. The processor has three-operand instructions that are:

ADD rX,rY,rZ	($rX \leftarrow rY+rZ$)
SUB rX,rY,rZ	($rX \leftarrow rY-rZ$)
MUL rX,rY,rZ	($rX \leftarrow rY \times rZ$)
DIV rX,rY,rZ	($rX \leftarrow rY/rZ$)
MOVE rX,rY	($rX \leftarrow rY$)

   Write the sequence of instructions corresponding to the desired calculation.

2. The processor has two-operand instructions that are:

ADD rX,rY	($rX \leftarrow rX+rY$)
SUB rX,rY	($rX \leftarrow rX-rY$)
MUL rX,rY	($rX \leftarrow rX \times rY$)
DIV rX,rY	($rX \leftarrow rX/rY$)
MOVE rX,rY	($rX \leftarrow rY$)

   Write the sequence of instructions corresponding to the desired calculation.

Exercise 4:

Here is a series of instructions:

```
        MVI   r1,#0 ; counter equal to 0
ici:    LDB   r2,(r0) ; get next character
        JZ    r2,fin ; end of the chain?
        ADD   r1,r1,#1 ; otherwise increment the counter
        ADD   r0,r0,#1 ; and go to the next character
        JMP   ici ; back in the loop
fin:
```

Based on the following instruction table, can you indicate what this program does?

Mnemonic	Operation	Meaning
MVI *rX*,#i	*rX* ← valeur i	The rX register receives on 32 bits the immediate value i indicated (MVI, MoVe Immediate).
LDB *rX*,(*rY*)	*rX* ← Mem[*rY*]$_8$	The register rX receives the memory byte whose address is in rY (LoaD Byte).
ADD *rX*,*rY*,*rZ* ADD *rX*,*rY*,#i	*rX* ← *rY* + *rZ* *rX* ← *rY* + valeur I	The rX register receives the 32-bit sum of the values contained in the rY and rZ registers (or the sum of rY and an immediate value i).
JMP Adr	PC ← Adr	Unconditional jump to the address Adr (JMP, JuMP).
JZ *rX*,Adr	PC ← Adr si *rX* = 0	Conditional jump to address Adr if register rX is zero (JZ, Jump if Zero).