Compilation: a bit about the target architecture

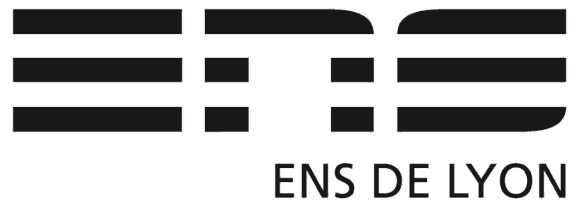
Laure Gonnord

http://laure.gonnord.org/pro/teaching/capM1.html

Laure.Gonnord@ens-lyon.fr

Master 1, ENS de Lyon

Sept 2017



Our target machine : LEIA

- Memory : 2^{16} 16-bits words.
- Instructions are also encoded in 16-bits words.
- Registers **PC**, **IR** + 16 general purpose registers **R0,...,R15**. We will use only the 8 first registers

We have a assembler and a simulator (see Lab 1)

LEIA ISA

See companion document.

Example : ADD instruction

- `add dr sr1 sr2`, does `dr <- sr1 + sr2`.
  - ↪ All operands are registers.
  - ↪ Example : `add r1 r2 r3` executes `r1 <- r2 + r3`.

- `add dr sr1 imm4`, does `dr <- sr + imm4`.
  - ↪ The last operand is an immediate value (on 4bits) encoded in the instruction.
  - ↪ Example : `add r1 r2 5` executes `r1 <- r2 + 5`.

LEIA ADD : encoding

A bit specifies the addressing mode :

class	action	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD reg	$r_d \leftarrow r_i + r_j$	0	0	0	1	**0**	d			i				j			
ADD imm4	$r_d \leftarrow r_i + \text{ext}(j)$	0	0	0	1	**1**	d			i				j			

Example : assemble `add r1 r2 5`

LEIA Memory access instructions

- "store" `wmem sr [dr]`, does `mem[dr] <- sr`.
- "load" `rmem dr [sr]`, does `dr <- mem[sr]`.

▶ **indirect addressing** : the address must be stored in a register `mem[rj]` the value at the address stored in r_j.

class	action	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
wmem	$\texttt{mem}[r_j] \leftarrow r_i$	0	0	0	0	0	0	0	0	i				j			
rmem	$r_d \leftarrow \texttt{mem}[r_j]$	1	1	1	1	d				0	0	0	0	j			

See also the `copy` instruction.

LEIA : branching

Unconditional branching :

- `jump c`, does `PC <- a + c` where a is the current instruction address.

class	action	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
jump	jump	1	0	1	1	c											

Test and branch :

- `snif r1 op r2`, skip the next instruction if the test is true

class	action	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
snif		0	0	1	1	1/0	condition			i				j			

Ex : Assembly code - demo

```
1   ;;; simple LEIA assembly demo
    leth r1 0                   ; first op
    letl r1 5

    .set r3 b                   ; second, from memory
6   rmem r2 [r3]

    add r4 r1 r2                ; add and store in memory
    .set r3 resu
    wmem r4 [r3]
11  jump 0                      ; stop

    b:
            .word 2
    resu:
16          .reserve 1          ; reservation of a memory cell
```

```
python3 asm.py demo.s && simu s demo.obj
```

LEIA Exercises

see TD sheet.