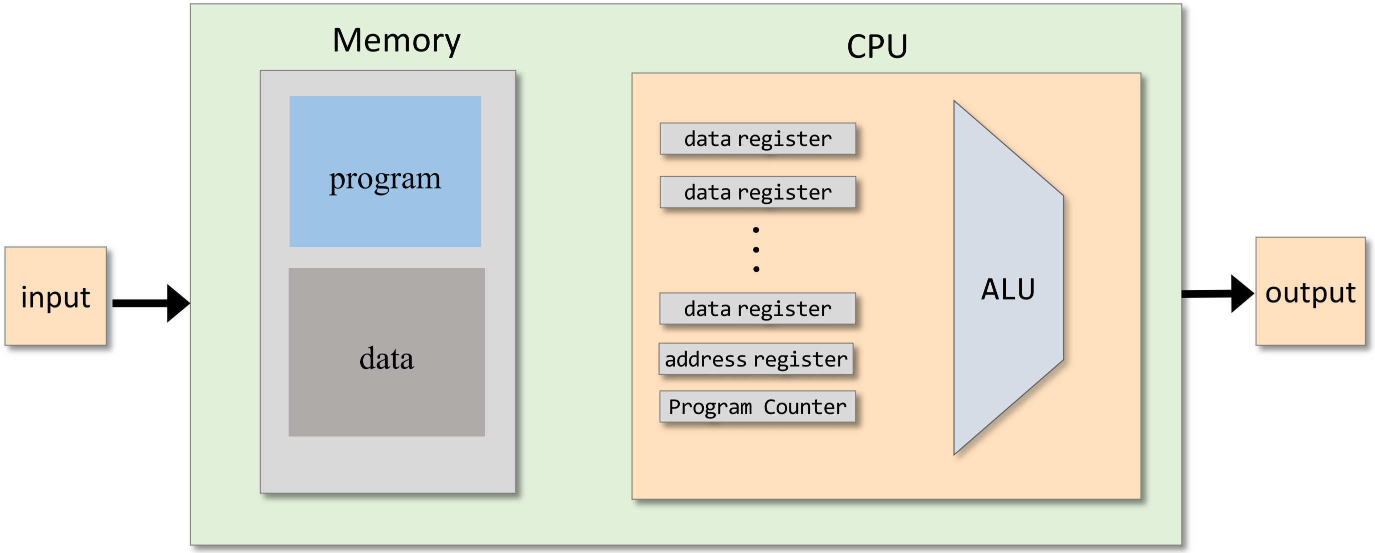
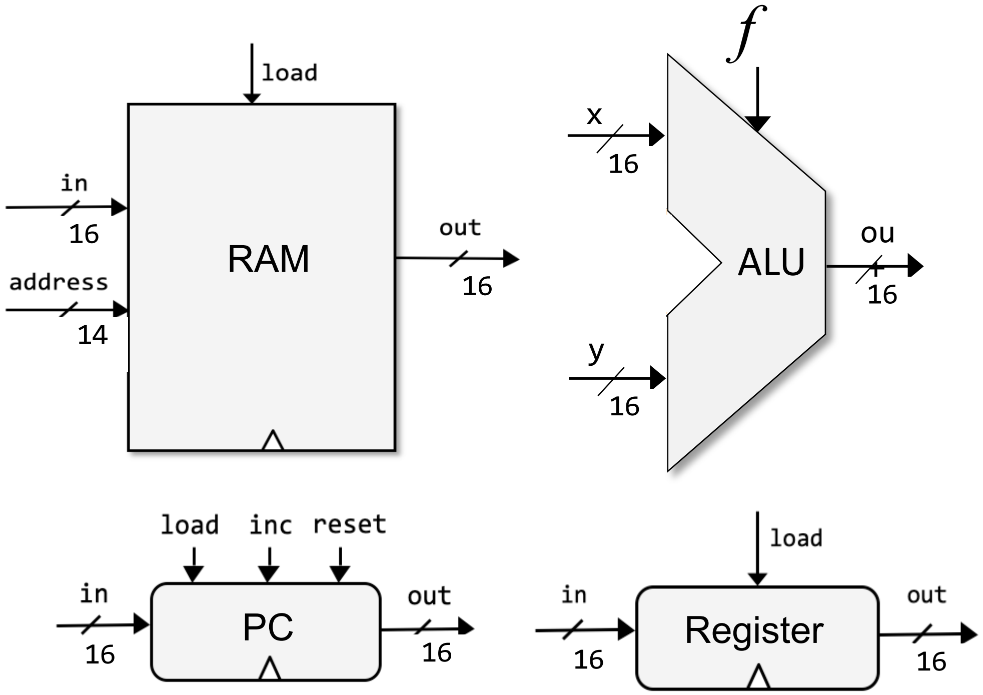
4. Architecture d’un ordinateur



Les données du problème

Composants matériels

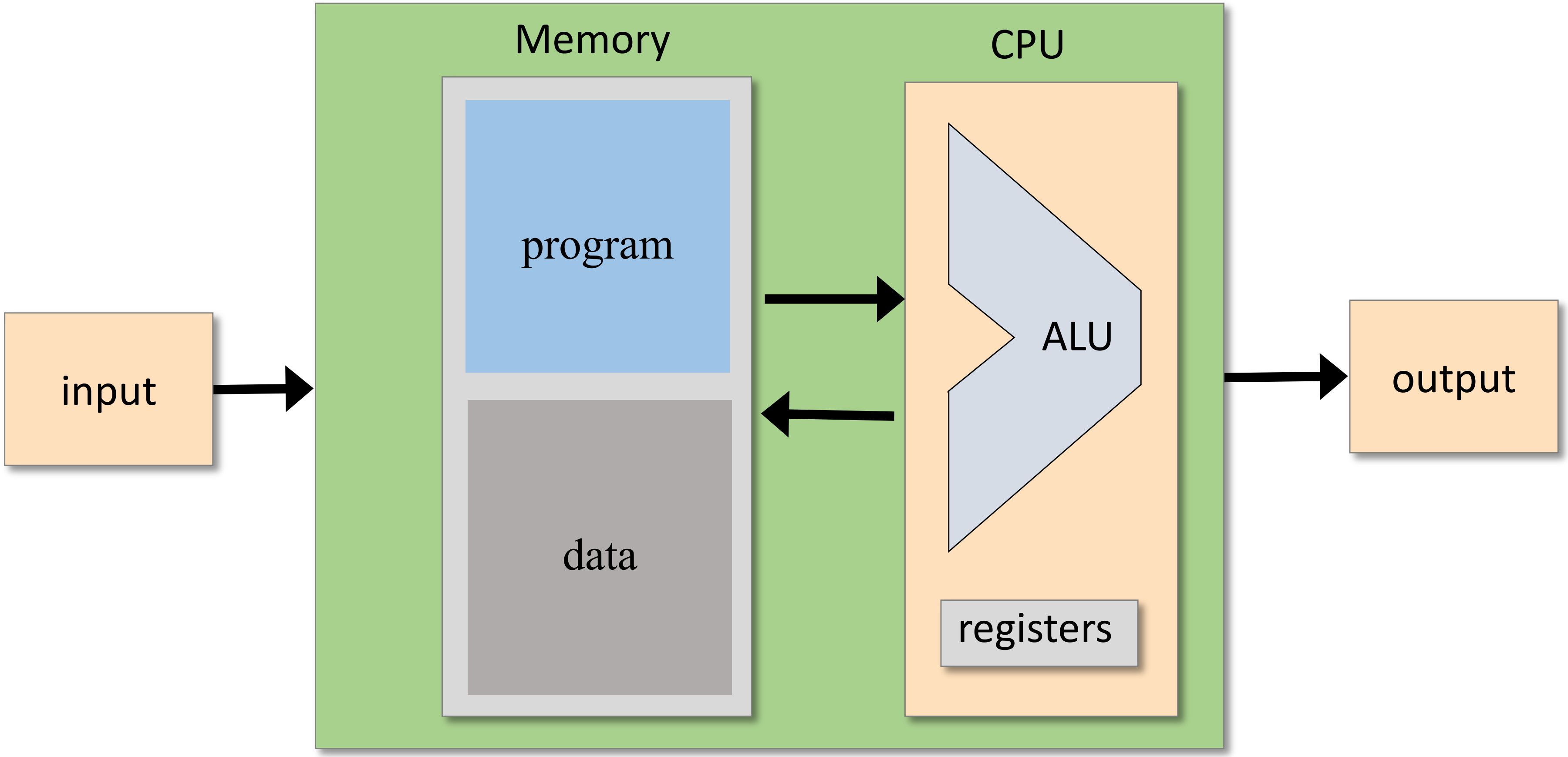


Langage machine

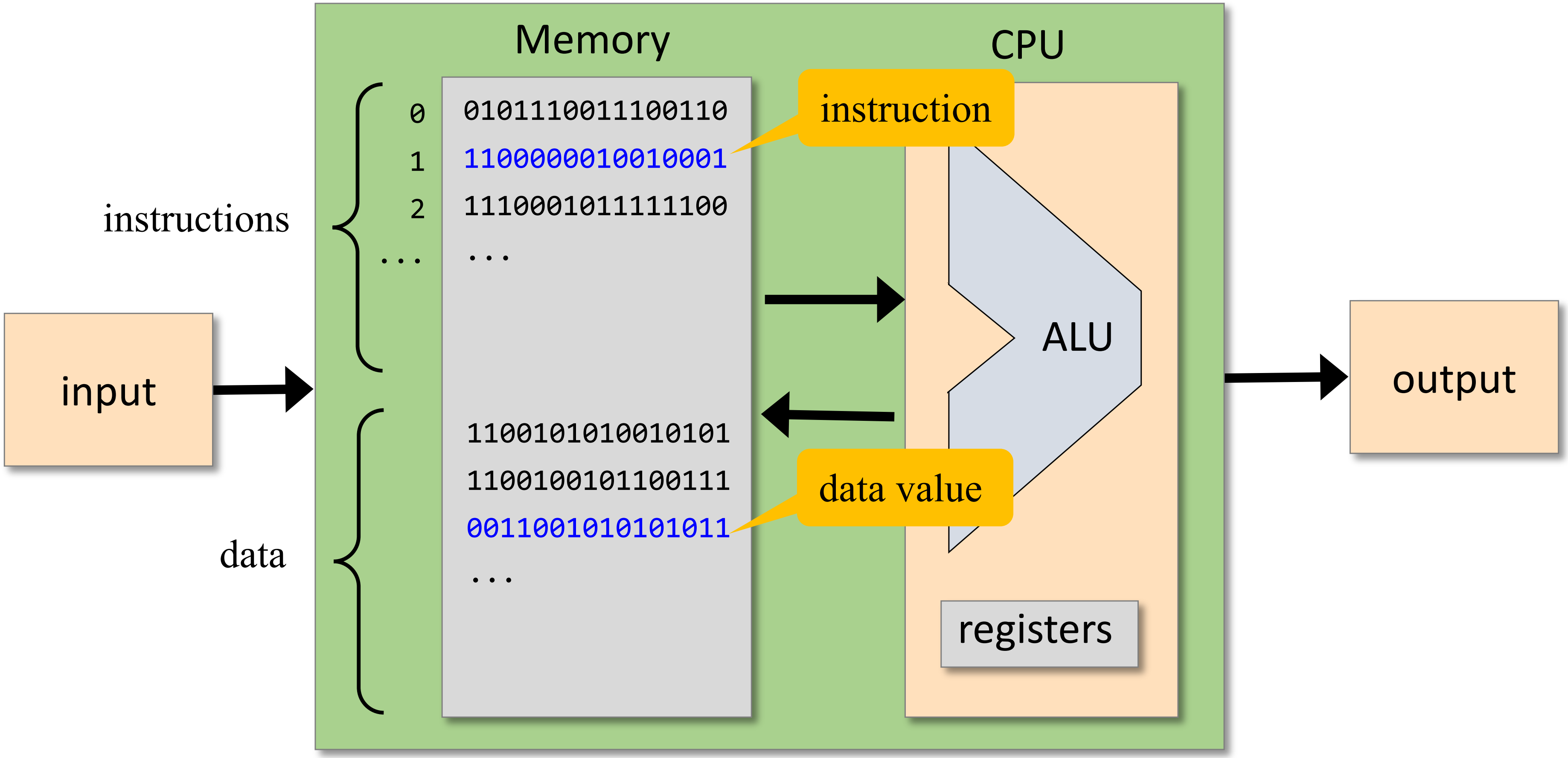
```
// Computes R1 = 1 + 2 + 3 + ... + R0
// i = 1
    @i
    M=1
    // sum = 0
    @sum
    M=0
(LOOP)
    // if (i > R0) goto STOP
    @i
    D=M
    @R0
    D=D-M
    @STOP
    D;JGT
    ...
```

Construire une machine pour l'exécution d'un programme !

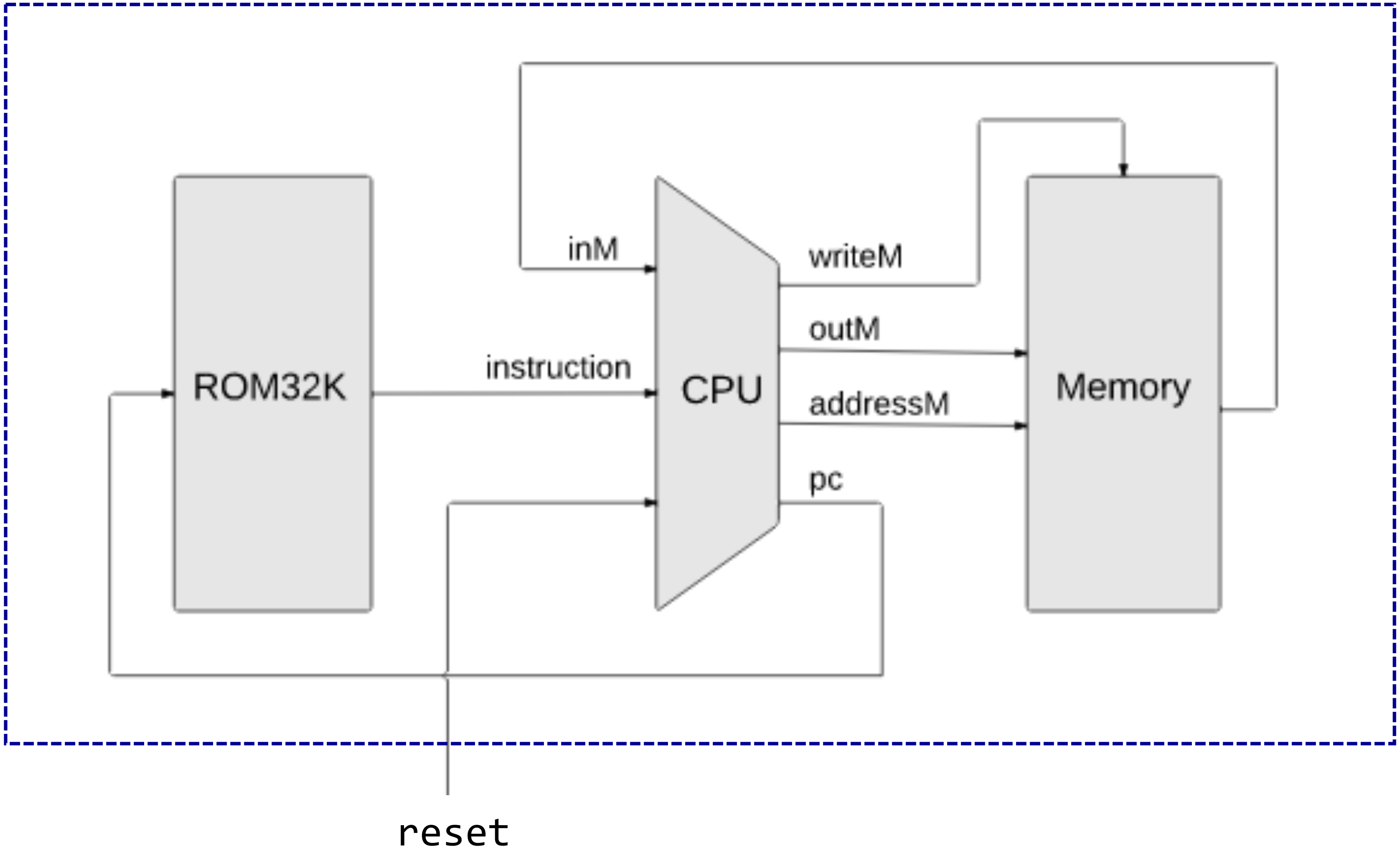
Architecture classique



Architecture classique



Implémentation de la Hack machine



Hack langage

A instruction

Symbolic: `@`*xxx* (*xxx* is a decimal value ranging from 0 to 32767, or a symbol bound to such a decimal value)

Binary: `0` *vvvvvvvvvvvvvvv* (*vv* ... *v* = 15-bit value of *xxx*)

C instruction

Symbolic: *dest* `=` *comp*`;` *jump* (*comp* is mandatory.
If *dest* is empty, the `=` is omitted;
If *jump* is empty, the `;` is omitted)

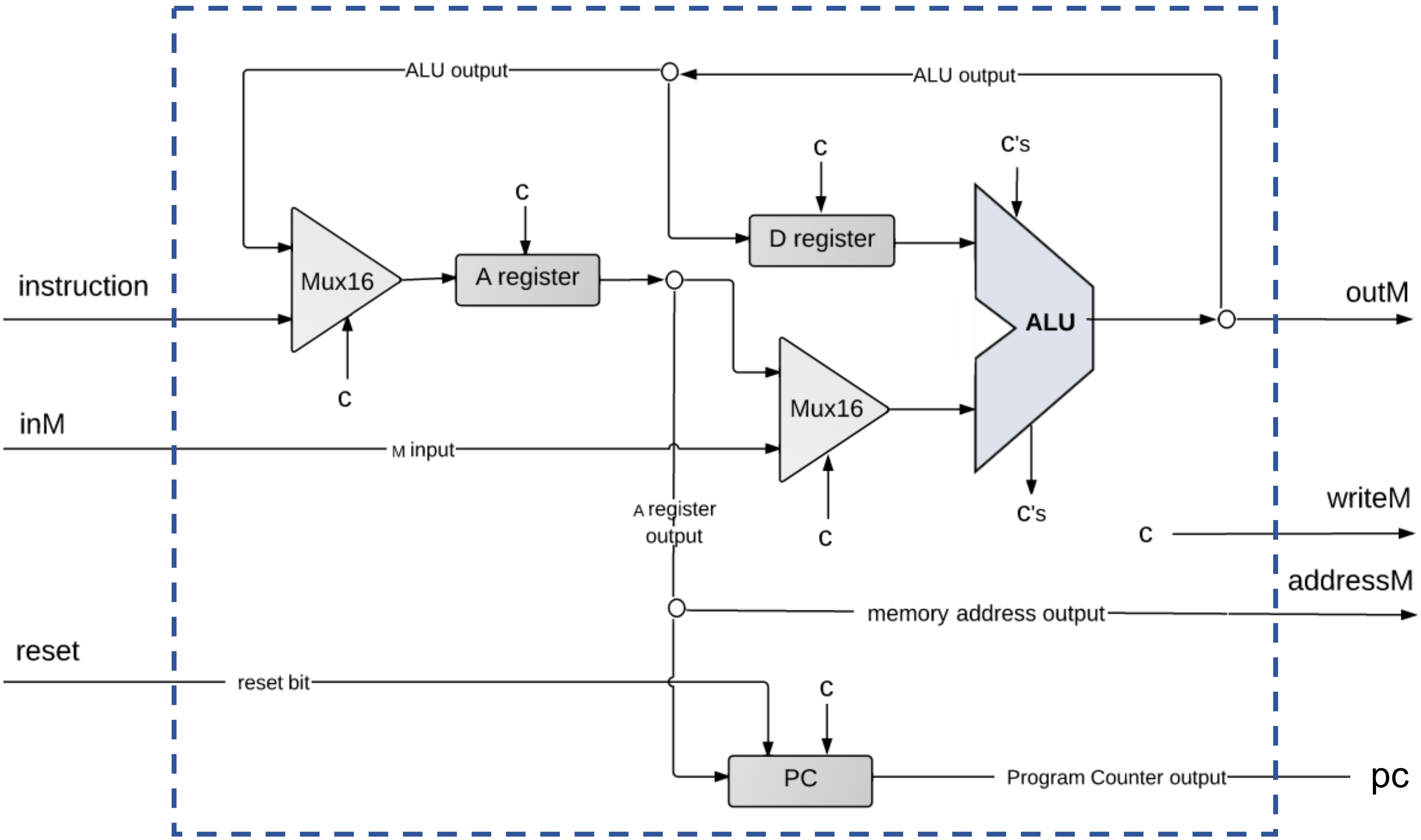
Binary: `111`*acccccccdddjjj*

comp		*c*	*c*	*c*	*c*	*c*	*c*
`0`		1	0	1	0	1	0
`1`		1	1	1	1	1	1
`-1`		1	1	1	0	1	0
`D`		0	0	1	1	0	0
`A`	`M`	1	1	0	0	0	0
`!D`		0	0	1	1	0	1
`!A`	`!M`	1	1	0	0	0	1
`-D`		0	0	1	1	1	1
`-A`	`-M`	1	1	0	0	1	1
`D+1`		0	1	1	1	1	1
`A+1`	`M+1`	1	1	0	1	1	1
`D-1`		0	0	1	1	1	0
`A-1`	`M-1`	1	1	0	0	1	0
`D+A`	`D+M`	0	0	0	0	1	0
`D-A`	`D-M`	0	1	0	0	1	1
`A-D`	`M-D`	0	0	0	1	1	1
`D&A`	`D&M`	0	0	0	0	0	0
`D|A`	`D|M`	0	1	0	1	0	1
a `== 0`	*a* `== 1`						

dest	*d*	*d*	*d*	Effect: store *comp* in:
`null`	0	0	0	the value is not stored
`M`	0	0	1	`RAM[A]`
`D`	0	1	0	D register (reg)
`DM`	0	1	1	`RAM[A]` and D reg
`A`	1	0	0	A reg
`AM`	1	0	1	A reg and `RAM[A]`
`AD`	1	1	0	A reg and D reg
`ADM`	1	1	1	A reg, D reg, and `RAM[A]`

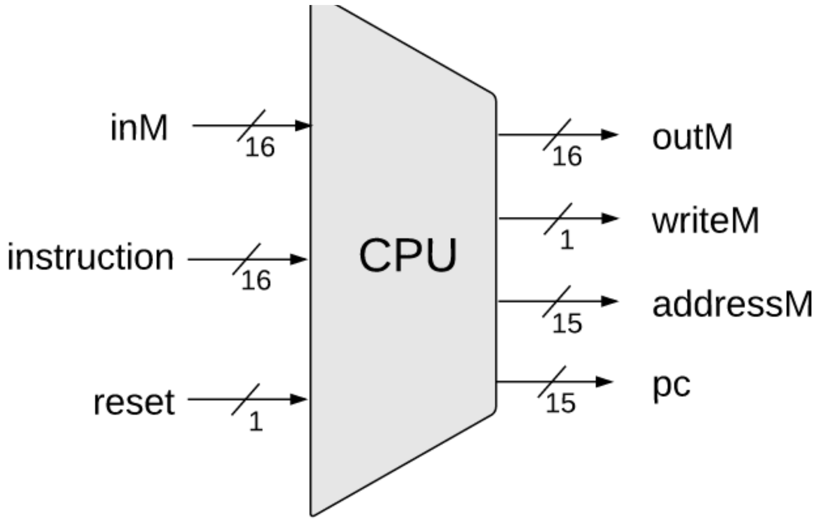
jump	*j*	*j*	*j*	Effect:
`null`	0	0	0	no jump
`JGT`	0	0	1	if *comp* $>$ 0 jump
`JEQ`	0	1	0	if *comp* $=$ 0 jump
`JGE`	0	1	1	if *comp* $\geq$ 0 jump
`JLT`	1	0	0	if *comp* $<$ 0 jump
`JNE`	1	0	1	if *comp* $\neq$ 0 jump
`JLE`	1	1	0	if *comp* $\leq$ 0 jump
`JMP`	1	1	1	unconditional jump

Implémentation de la CPU de la Hack machine



Exercices : Calculer les fonctions de contrôles des Mux, registres et ALU.

HDL de la CPU



```
/** Central Processing unit.
    Executes instructions written in Hack machine language.
CHIP CPU {

   IN
      inM[16],          // Value of M  (RAM[A])
      instruction[16],  // Instruction to execute
      reset;            // Signals whether to execute the first instruction
                        // (reset==1) or next instruction (reset == 0)

    OUT
       outM[16]         // Value to write to the selected RAM register
       writeM,          // Write to the RAM?
       addressM[15],    // Address of the selected RAM register
       pc[15];          // Address of the next instruction

    PARTS:
    // Put you code here:
}
```

Questions