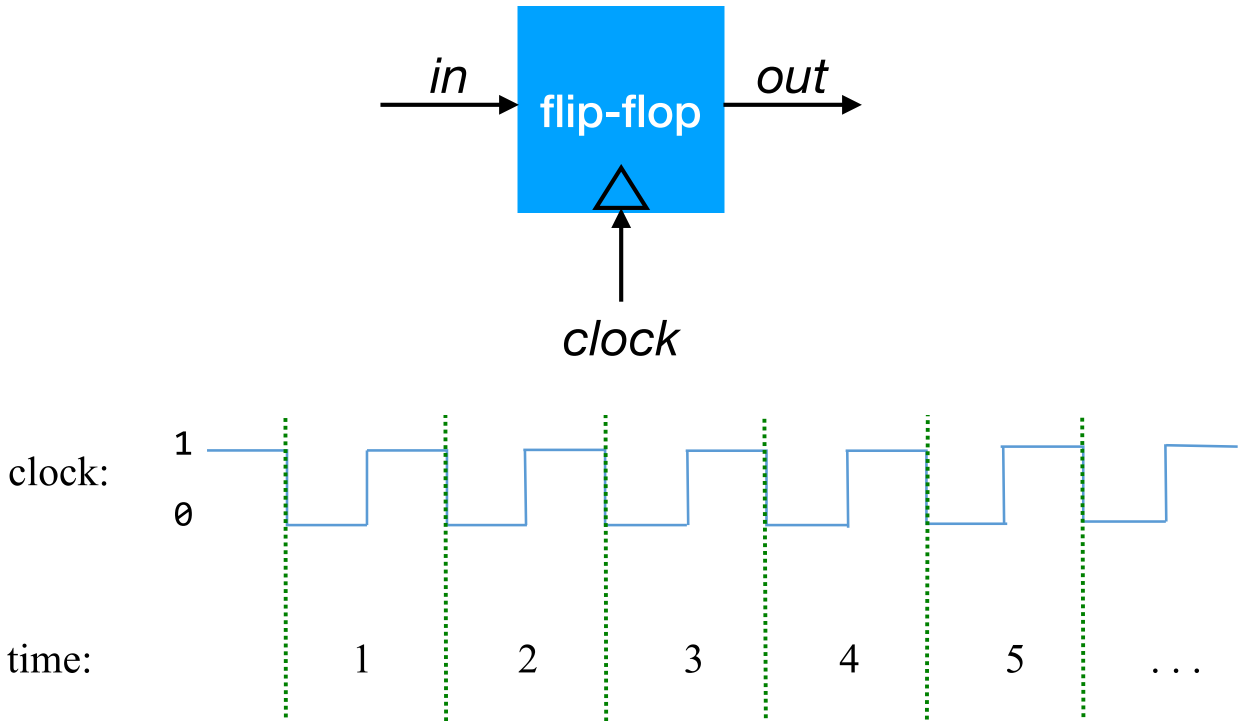
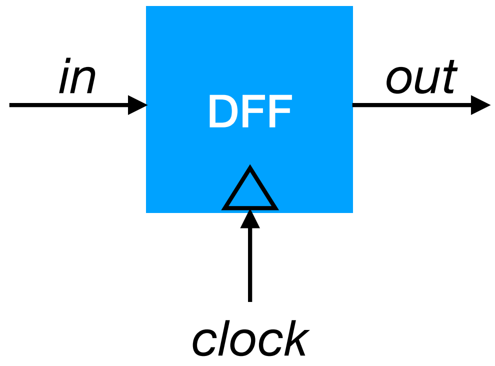
3. Logique séquentielle



Data Flip-Flop



t	0	1	2	3	4	5	6	7	8	9	…
in	0	0	1	0	1	1	1	0	1	0	…
out	?	0	0	1	0	1	1	1	0	1	0

Problème : Comment décrire formellement le comportement du DFF ?

Data Flip-Flop

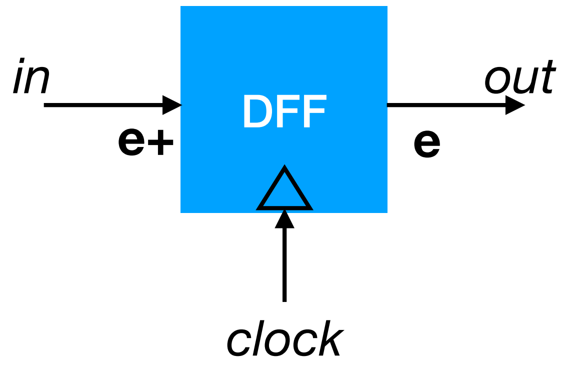
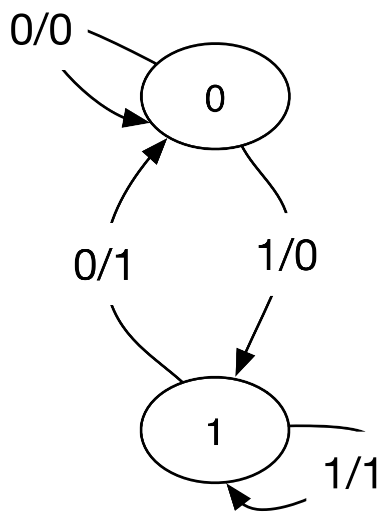


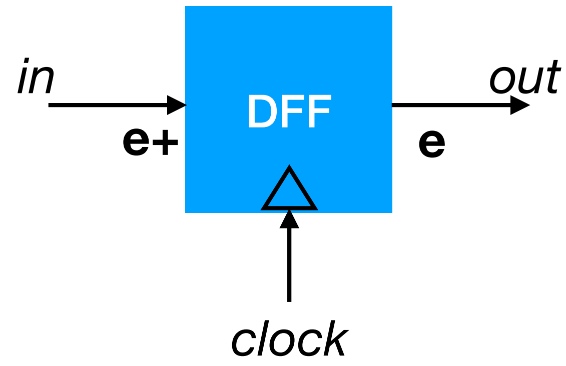
Table de vérité

in	e	e+	out
0	**0**	0	0
0	**1**	0	1
1	**0**	1	0
1	**1**	1	1

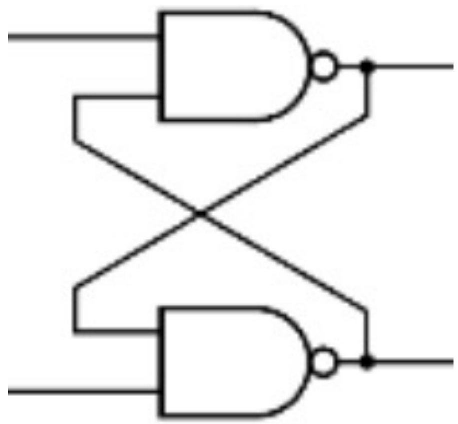
Machine à états



HDL d’un Data Flip-Flop




```
/** Data Flip-flop: out(t) = in(t–1)
 *   where t is the current time unit. */

CHIP DFF {
    IN  in;
    OUT out;

    BUILTIN DFF;
    CLOCKED in;
}
```



Extrait du circuit d’un DFF

Forme générique d’un circuit séquentiel

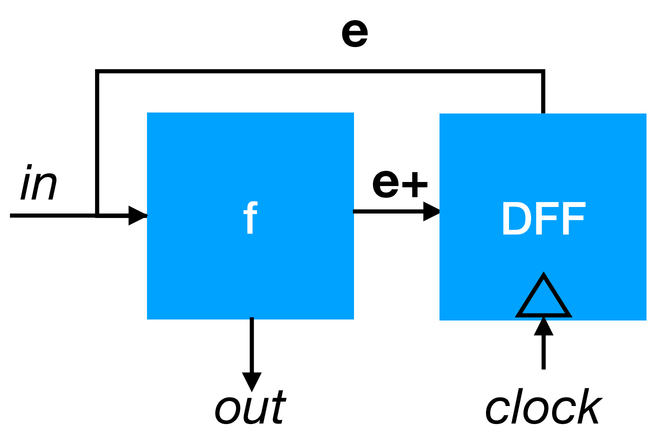


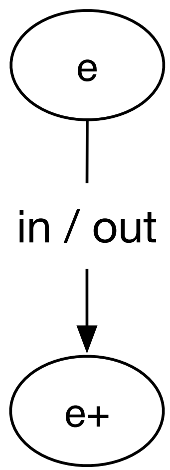
Table de vérité

f(in, e) = e+,out

in	e	e+	out

une ligne = une transition

Machine à états

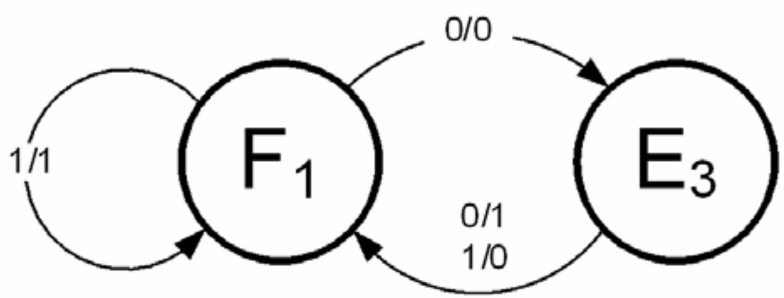


Exercices

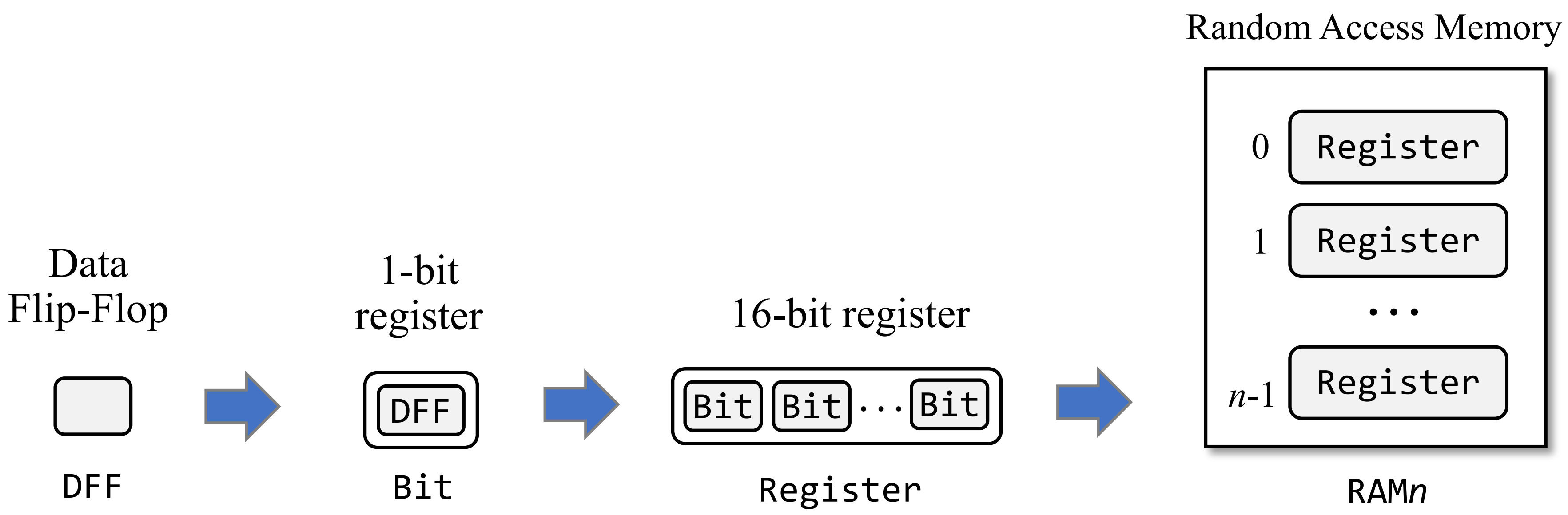
- Donner la table de vérité et la machine à état du composant Machin

```
CHIP Machin {
     IN in, load;
     OUT out;

     PARTS:
Mux(a=gayout,b=in,sel=load,out=a);
DFF(in=a,out=out,out=gayout);
}
```

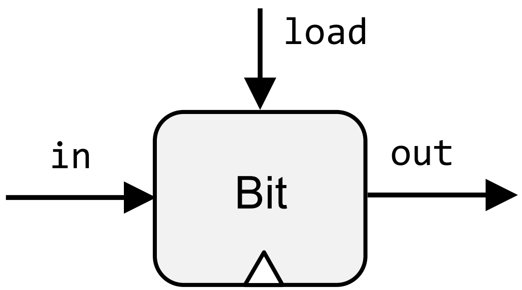
- Donner la table de vérité et le code HDL décrit par la machine état suivante

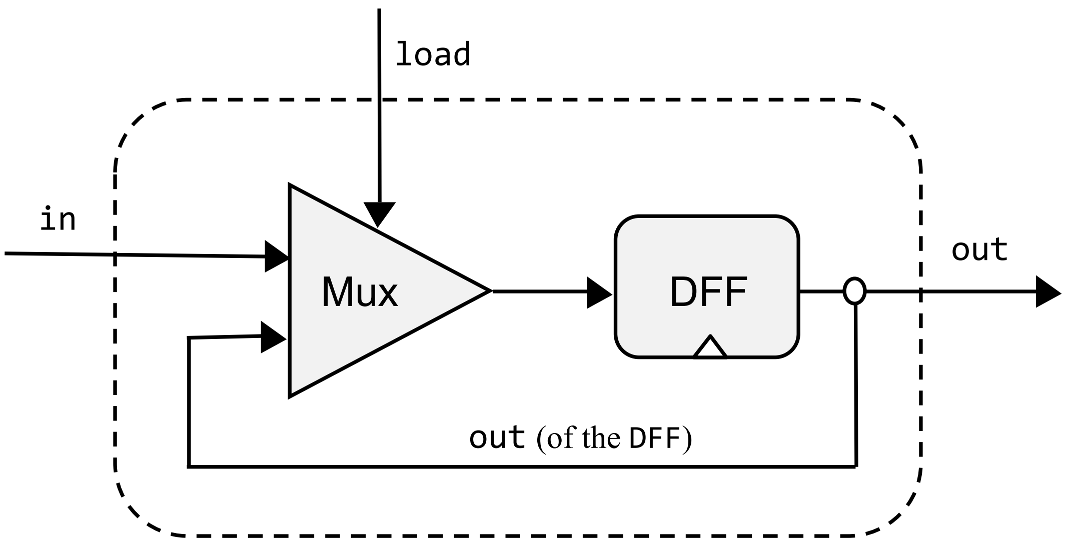


Du DFF à la RAM



Registre 1 bit



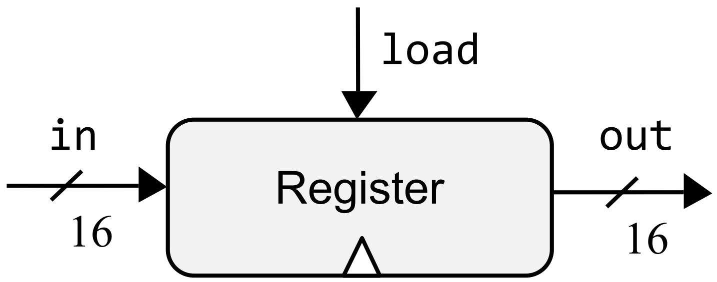


Bit.hdl

```
/** 1-Bit register:
    if  load(t) then out(t + 1) = in(t)
    else              out(t + 1) = out(t)  */

CHIP Bit {
    IN in, load;
    OUT out;

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

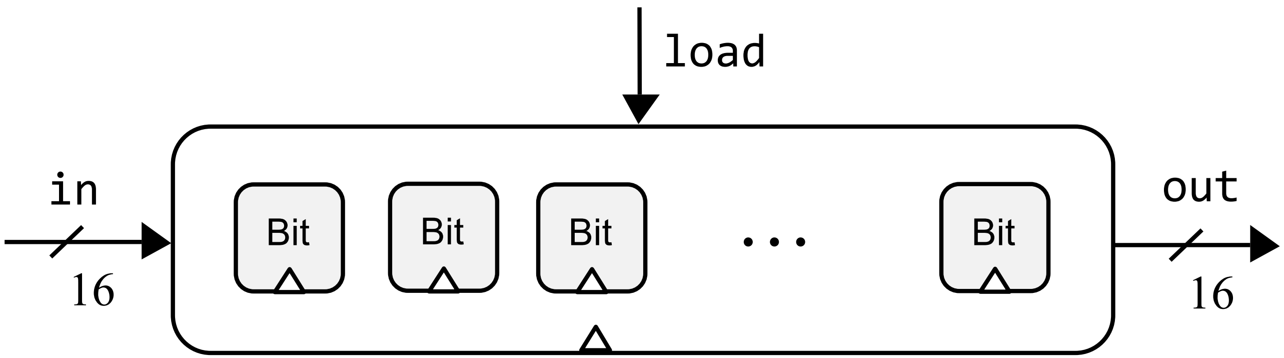
Exercice : écrire la table de vérité et donner la machine à états

Registre 16 bits



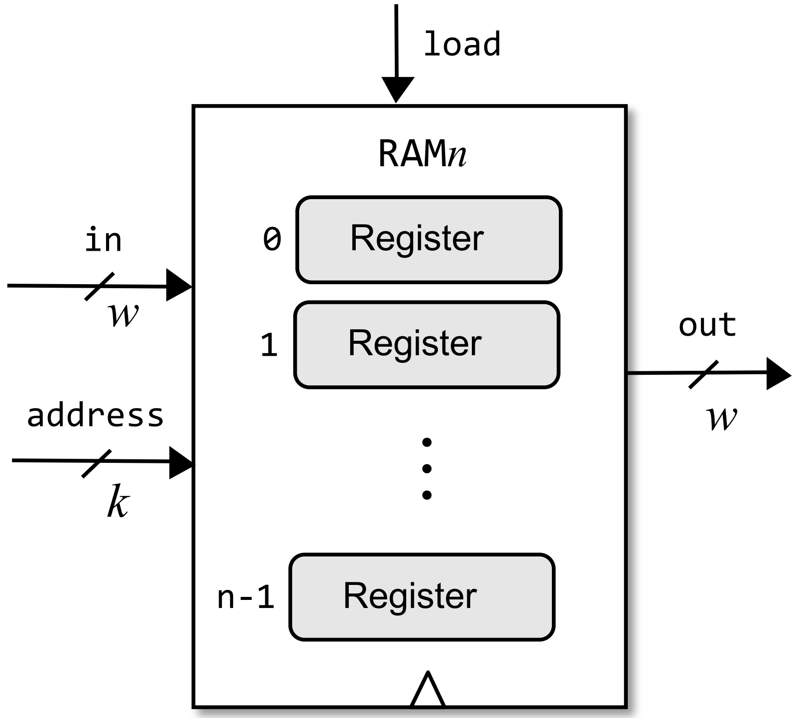
Register.hdl

```
/** 16-bit register:
    if  load(t) then out(t + 1) = in(t)
    else             out(t + 1) = out(t)  */

CHIP Register{
    IN in[16], load;
    OUT out[16];

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



Exercice : compléter le code HDL du registre 16 bits

Random Access Memory (RAM)



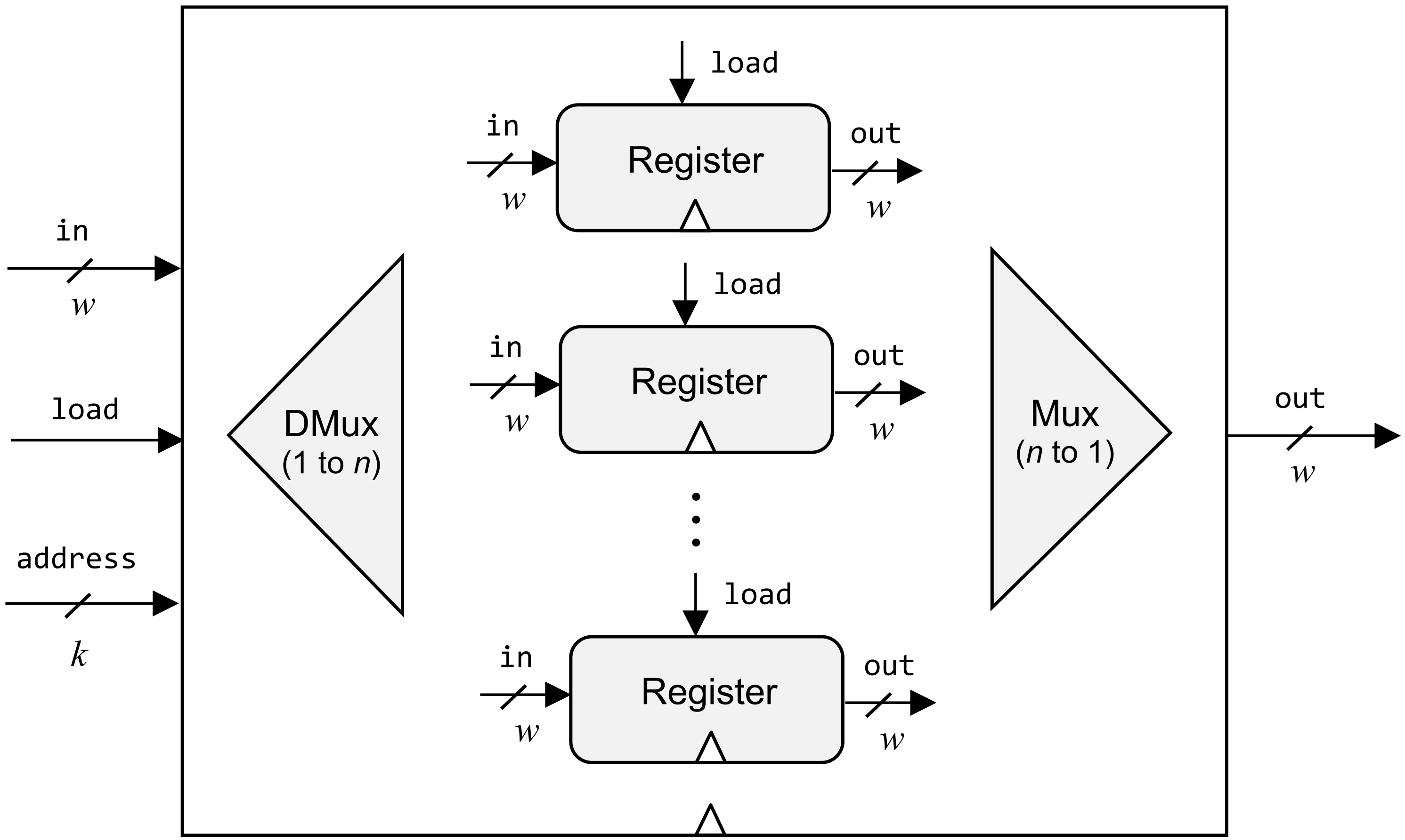
Comportement

- si load = 0 alors la RAM ne change pas d'état
- si load = 1 alors RAM[address] = in

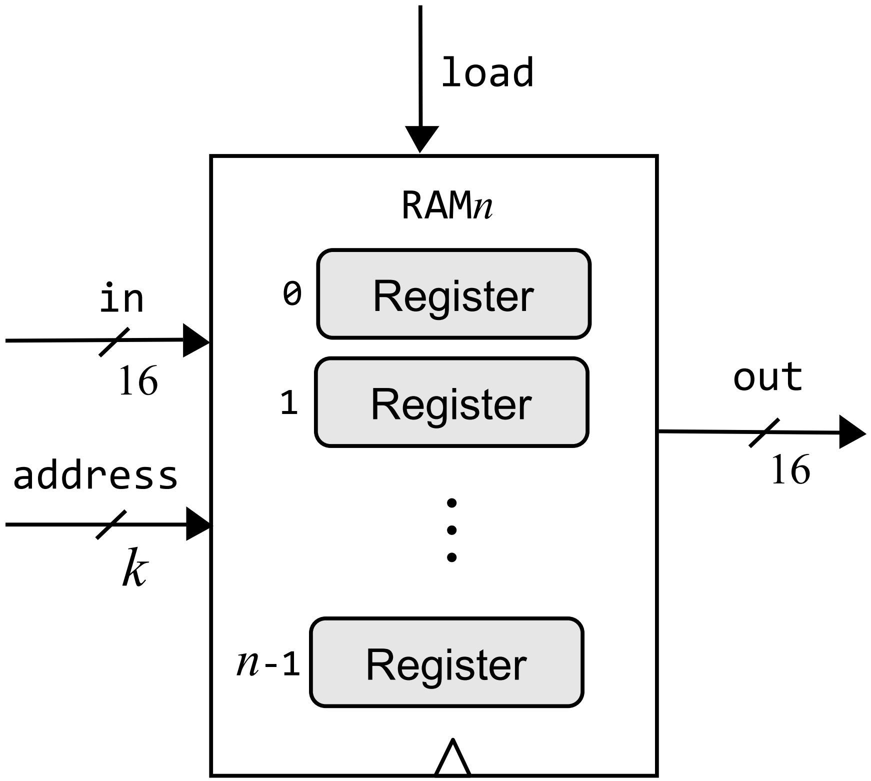
- out = RAM[address] avec un temps de retard !

Exercice : construire une « RAM 16 bits de taille 8 »

Principe de la construction d'une RAM



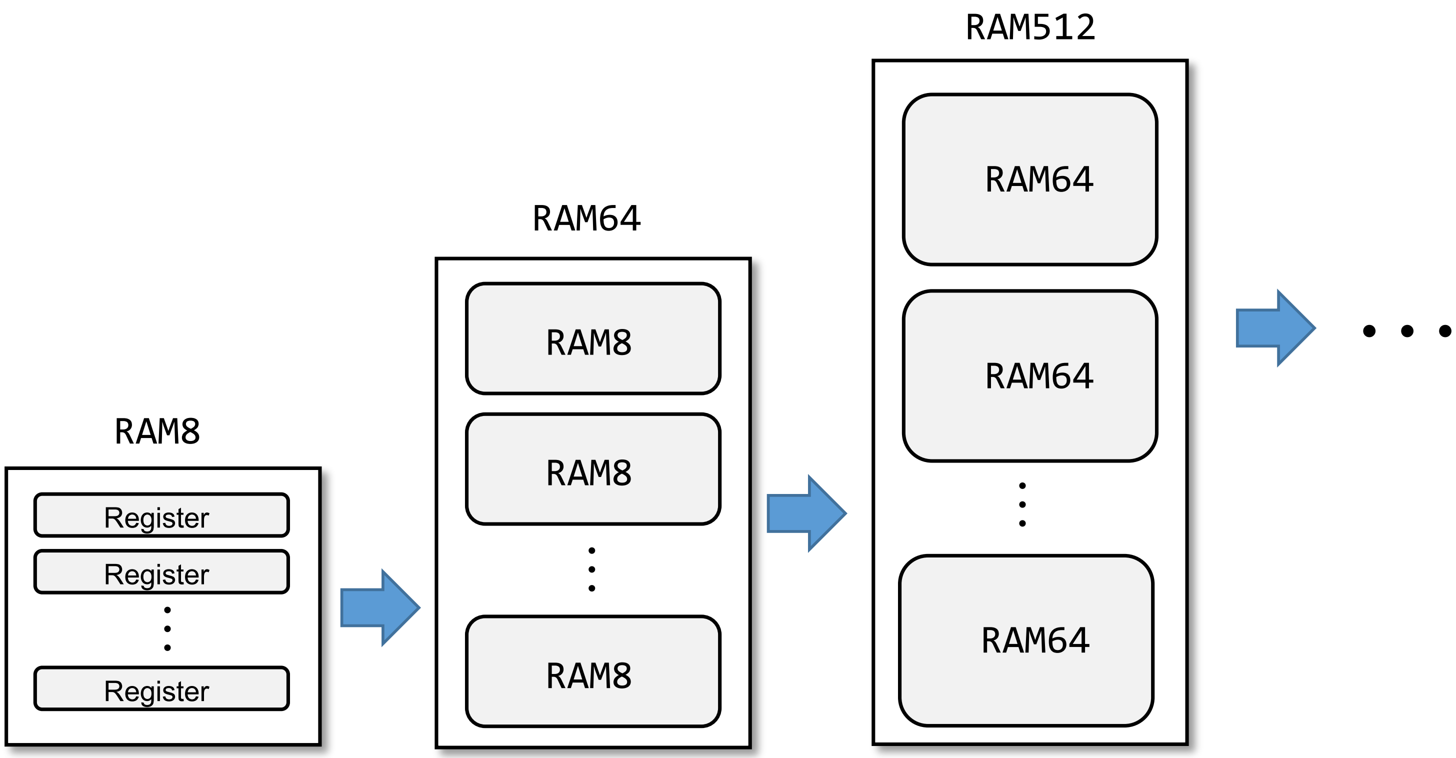
RAM de la Hack Machine



chip name	n	k
RAM8	8	3
RAM64	64	6
RAM512	512	9
RAM4K	4096	12
RAM16K	16384	14

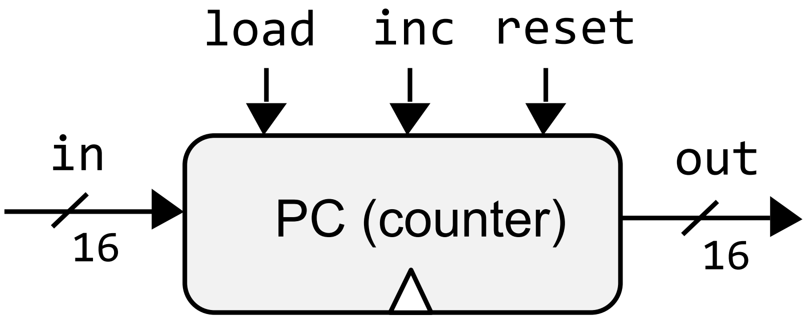
A savoir : 1K = 2^{10} =1024 ; 1M = 2^{10} K = 2^{20} ; 1G = 2^{30} K ; 1T = 2^{40}

Technique de construction des RAM



Exercices : construire les RAM de la Hack machine (RAM64, RAM512, RAM4K, RAM16K)

Program counter (PC)



if `reset`(t)	`out`$(t+1)$ = `0`
else if `load`(t)	`out`$(t+1)$ = `in`(t)
else if `inc`(t)	`out`$(t+1)$ = `out`(t) + `1`
else	`out`$(t+1)$ = `out`(t)

Trois opérations élémentaires sur le compteur ordinal (PC)

- <u>Reset</u> : récupérer la première instruction PC = 0
- <u>Next</u> : récupérer l’instruction suivante PC++
- <u>Goto</u> : récupérer la nième instruction PC = n

HDL du Program counter (PC)

```
/**
 * A 16-bit counter with load and reset control bits.
 * if      (reset[t]==1) out[t+1] = 0
 * else if (load[t]==1)  out[t+1] = in[t]
 * else if (inc[t]==1)   out[t+1] = out[t] + 1
(integer addition)
 * else                  out[t+1] = out[t]
 */

CHIP PC {
    IN in[16],load,inc,reset;
    OUT out[16];

    PARTS:

}
```

Exercice : construire en HDL le « program counter »

Nos composants de la Hack Machine

Etant donné :

- DDF

Nous savons construire :

- Bit
- Register
- PC
- RAM8
- RAM64
- RAM512
- RAM4K
- RAM16K

Questions