COM-ORG

Project-3 Overview

31110321 Computer Organization
สำหรับนักศึกษาชั้นปีที่ 3 สาขาวิชาวิศวกรรมคอมพิวเตอร์

ทรงฤทธิ์ กิติศรีวรพันธุ์

songrit@npu.ac.th

สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยนครพนม

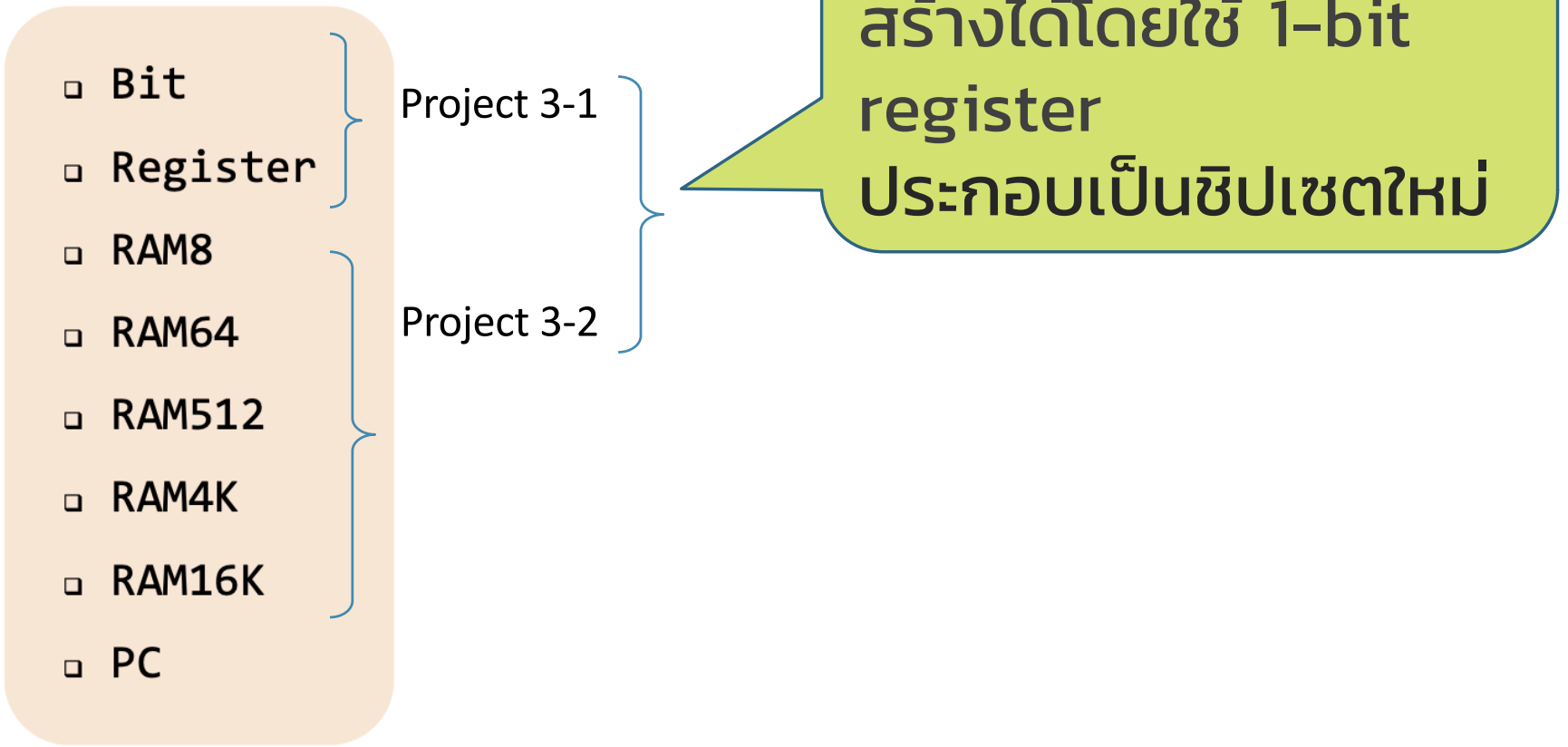
Revised 2020-02-15

Text book

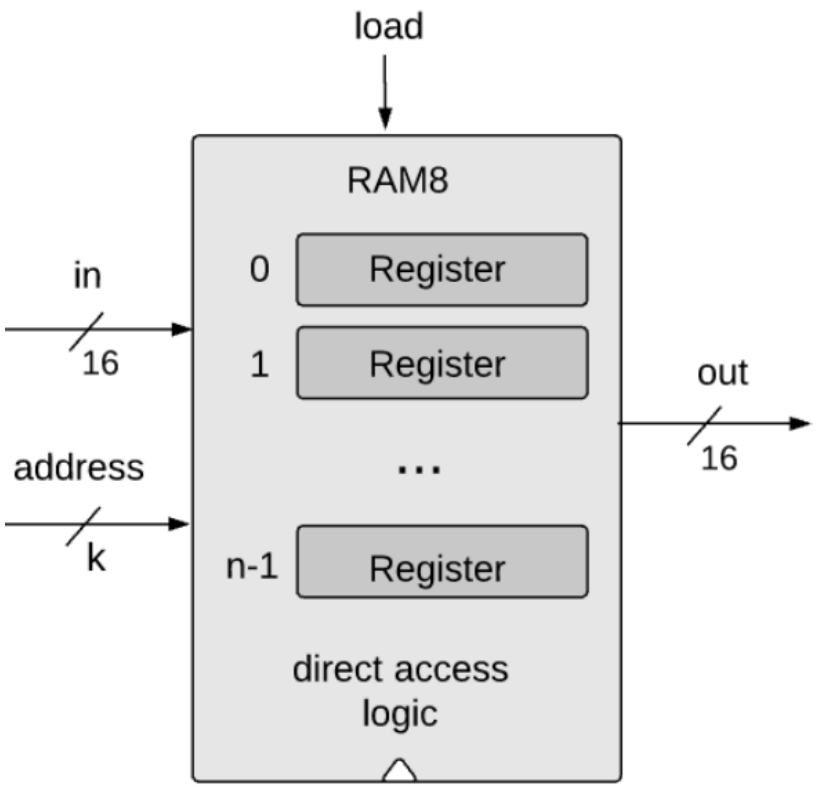
- https://files.npu.world/321-comorg/
  - บทที่ 1 ch1.pdf
  - บทที่ 2 ch2.pdf
  - บทที่ 3 ch3.pdf

Outline

- Chipset : ใช้ชิปจาก Project-1, 2
- Goal : สร้างชิปเซตต่อไปนี้

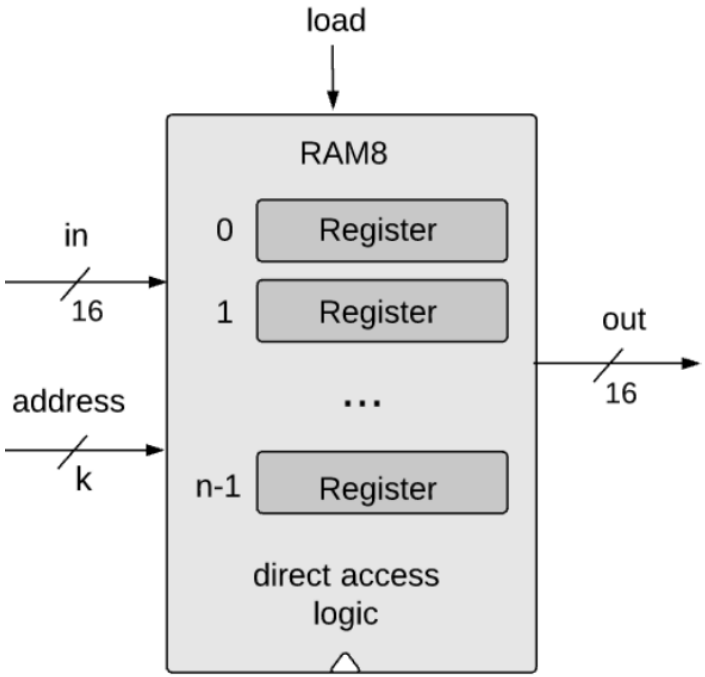


แนะนำหน่วยความจำ



- สถาปัตยกรรม
  - ทำงานเป็นลำดับ
  - เริ่มจาก address 0 ไป n-1
- จำนวน Address width:
  - $k = \log_2 n$
- Word width
  - Hack computer , w=16
  - โทรศัพท์มือถือ *w=64*
  - *คอมพิวเตอร์ w=64*
  - *เครื่องเล่นเกม famicom w=8*

RAM



- การติดต่อ RAM
- RAM เป็นชิปรวม Register หลายตัว
- การติดต่อ RAM แต่ละครั้งเป็นการติดต่อ Register แต่ละตัวภายในชิป
- การติดต่อ 1 ครั้ง อ่านชิปได้ 1 ตัว
- ตำแหน่งชิปเรียกว่า Address

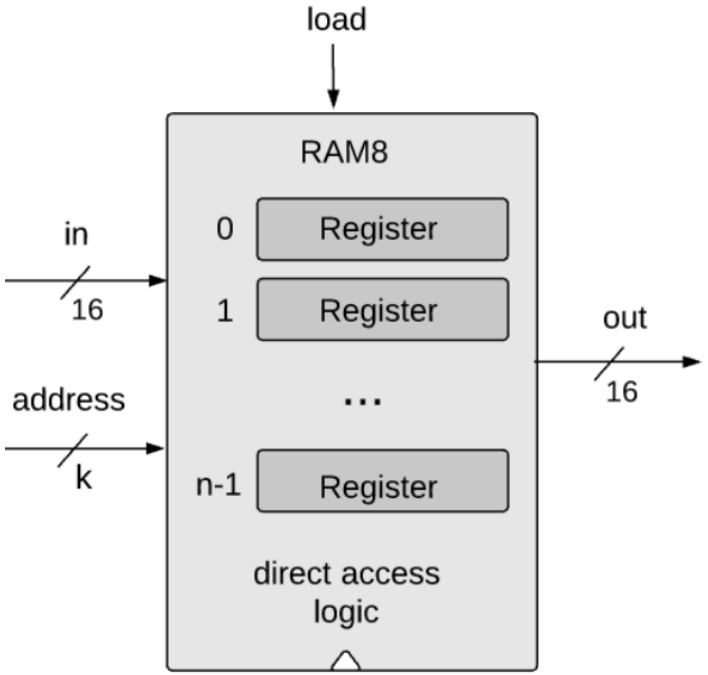
อ่าน หน่วยความจำ ตำแหน่ง address=1

set address=1
probe outs

เขียน 5 ลงหน่วยความจำ address = 2

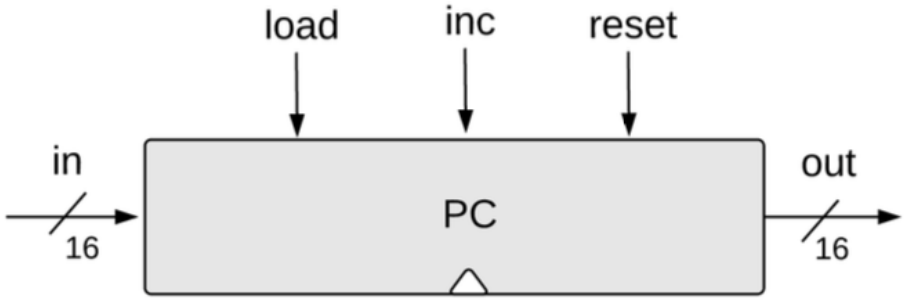
set address = 2
set in=5
set load=1

Random Access Memory



- อ่าน register ภายในแรม
- ไม่ว่าตำแหน่งใดก็ตาม
- ใช้เวลาไม่น้อยไม่มากไปกว่ากัน

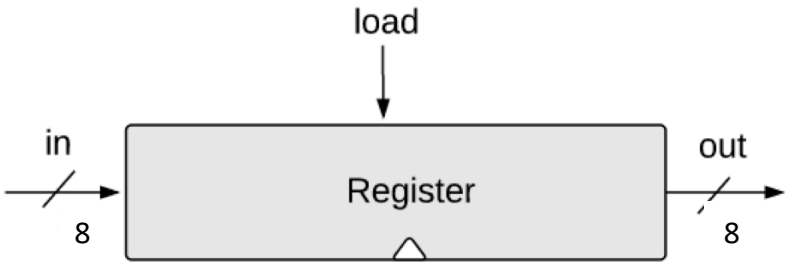
Program Counter (PC)




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

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

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

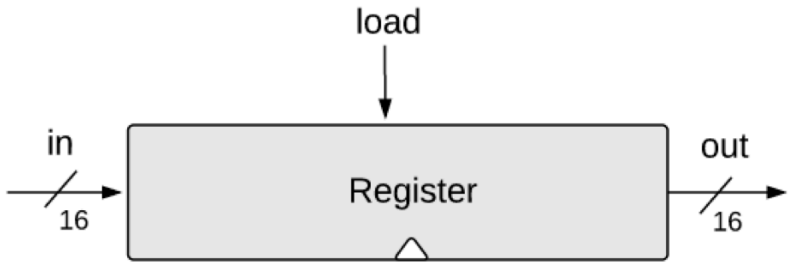
```
CHIP PC {
  IN in[16],load,inc,reset;
  OUT out[16];

PARTS:
   Inc16(in=oo,out=incd);
   Mux16(a=oo,b=incd,sel=inc,out=o);
   Mux16(a=o,b=in,sel=load,out=uu);
   Mux16(a=uu,b[0..15]=false, sel=reset,out=this);
   Register(in=this,load=true,out=out,out=oo);
}
```

8-bit register




```
/**
 * 8-bit register:
 * If load(t) then out(t+1) = in(t)
 * else            out(t+1) = out(t))
 */
CHIP 8Bit {
   IN in[8], load;
   OUT out[8];

   PARTS:
    // Put your code head:

}
```

16-bit register



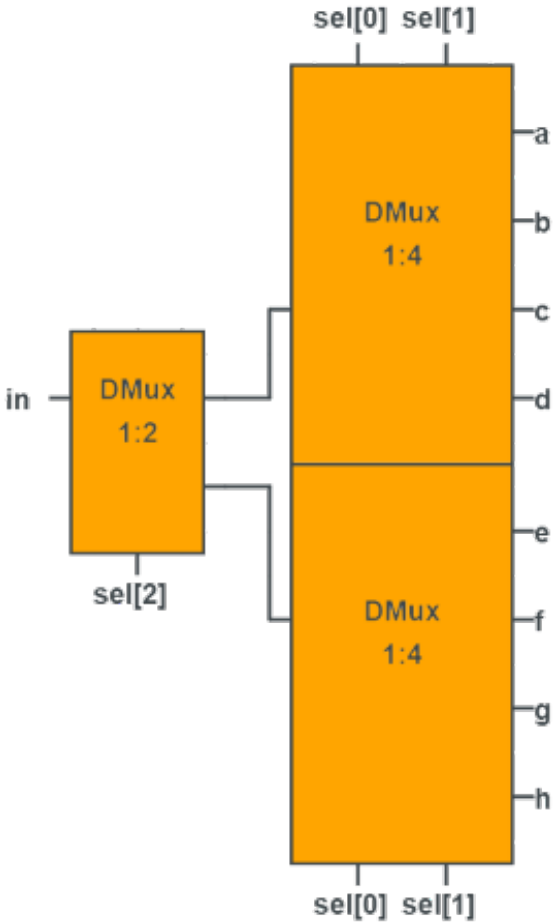
- สร้าง 16-bit register โดยใช้
  - 1-bit register

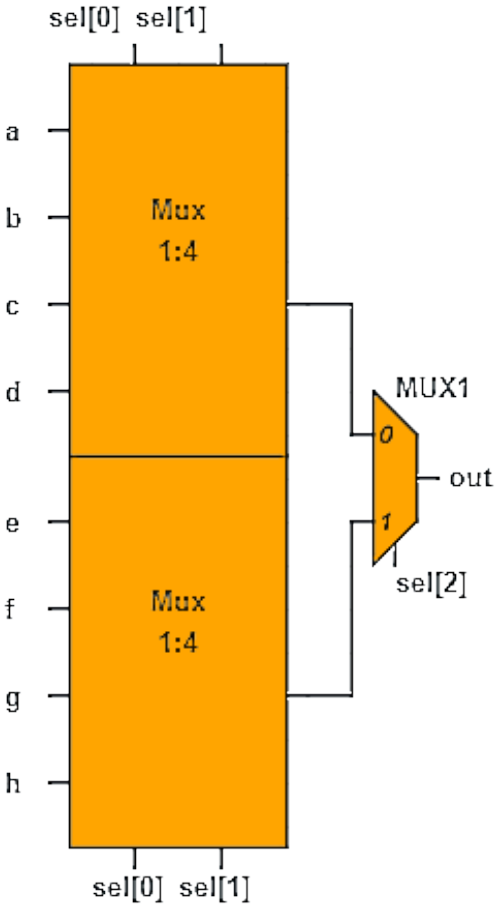
Register.hdl

```
/**
 * 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:
}
```

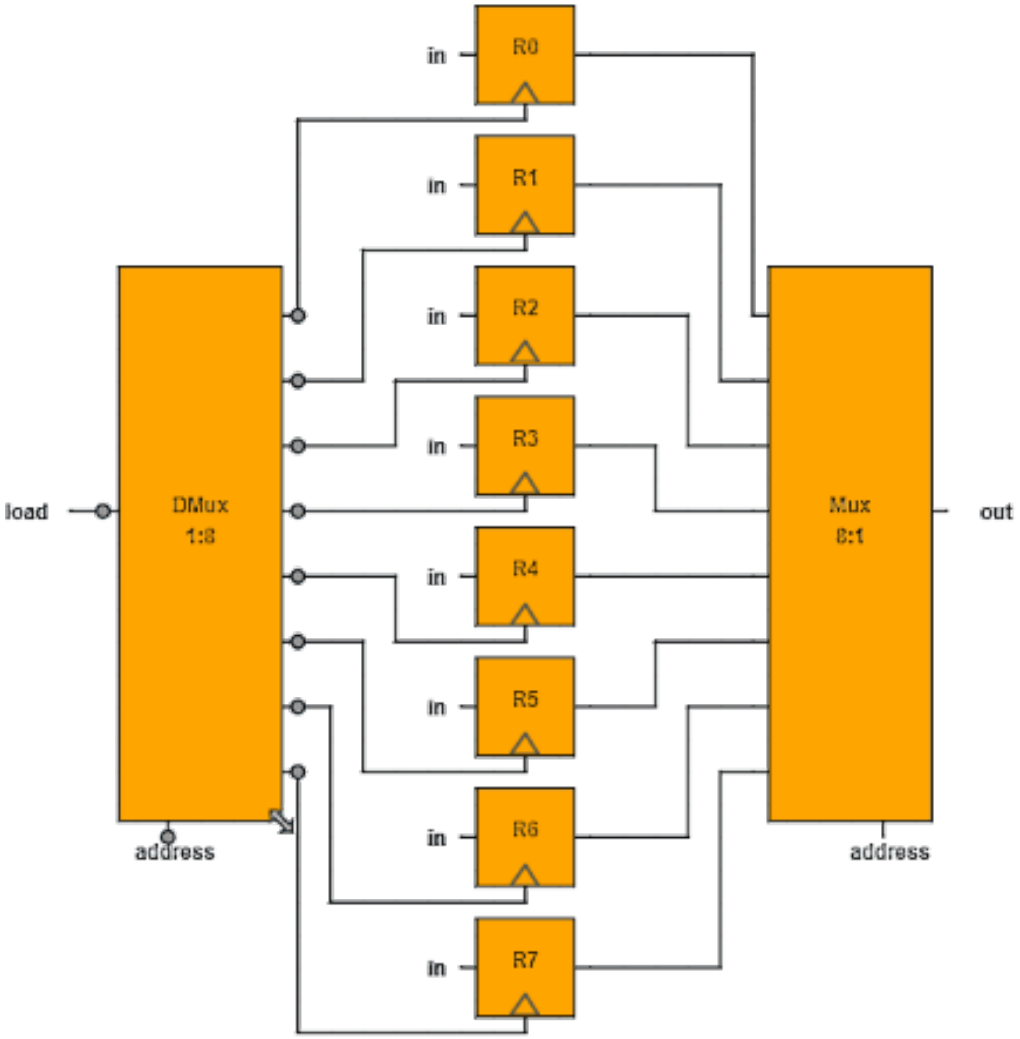
```
CHIP Register {
   IN in[16], load;
   OUT out[16];

   PARTS:
        Bit(in=in[0],load=load,out=out[0]);
        Bit(in=in[1],load=load,out=out[1]);
        Bit(in=in[2],load=load,out=out[2]);
        Bit(in=in[3],load=load,out=out[3]);
        Bit(in=in[4],load=load,out=out[4]);
        Bit(in=in[5],load=load,out=out[5]);
        Bit(in=in[6],load=load,out=out[6]);
        Bit(in=in[7],load=load,out=out[7]);
        Bit(in=in[8],load=load,out=out[8]);
        Bit(in=in[9],load=load,out=out[9]);
        Bit(in=in[10],load=load,out=out[10]);
        Bit(in=in[11],load=load,out=out[11]);
        Bit(in=in[12],load=load,out=out[12]);
        Bit(in=in[13],load=load,out=out[13]);
        Bit(in=in[14],load=load,out=out[14]);
        Bit(in=in[15],load=load,out=out[15]);
}
```

Dmux8Way แล: Mux8Way

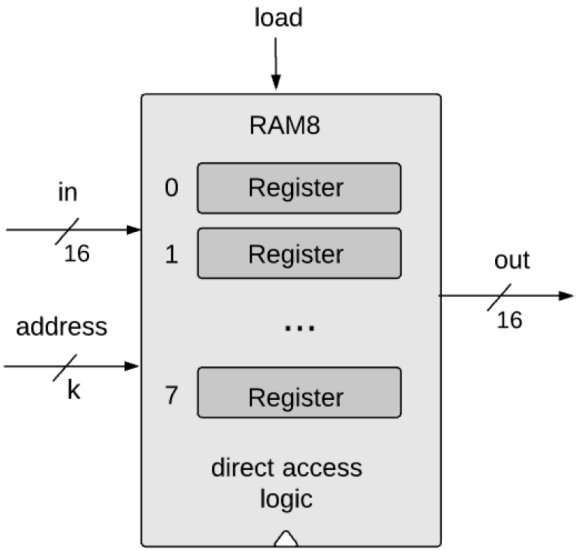




Ram8


in
R0
in
R1
in
R2
in
R3
in
R4
in
R5
in
R6
in
R7
load
DMux
1:8
address
Mux
8:1
out
address

8-Register RAM



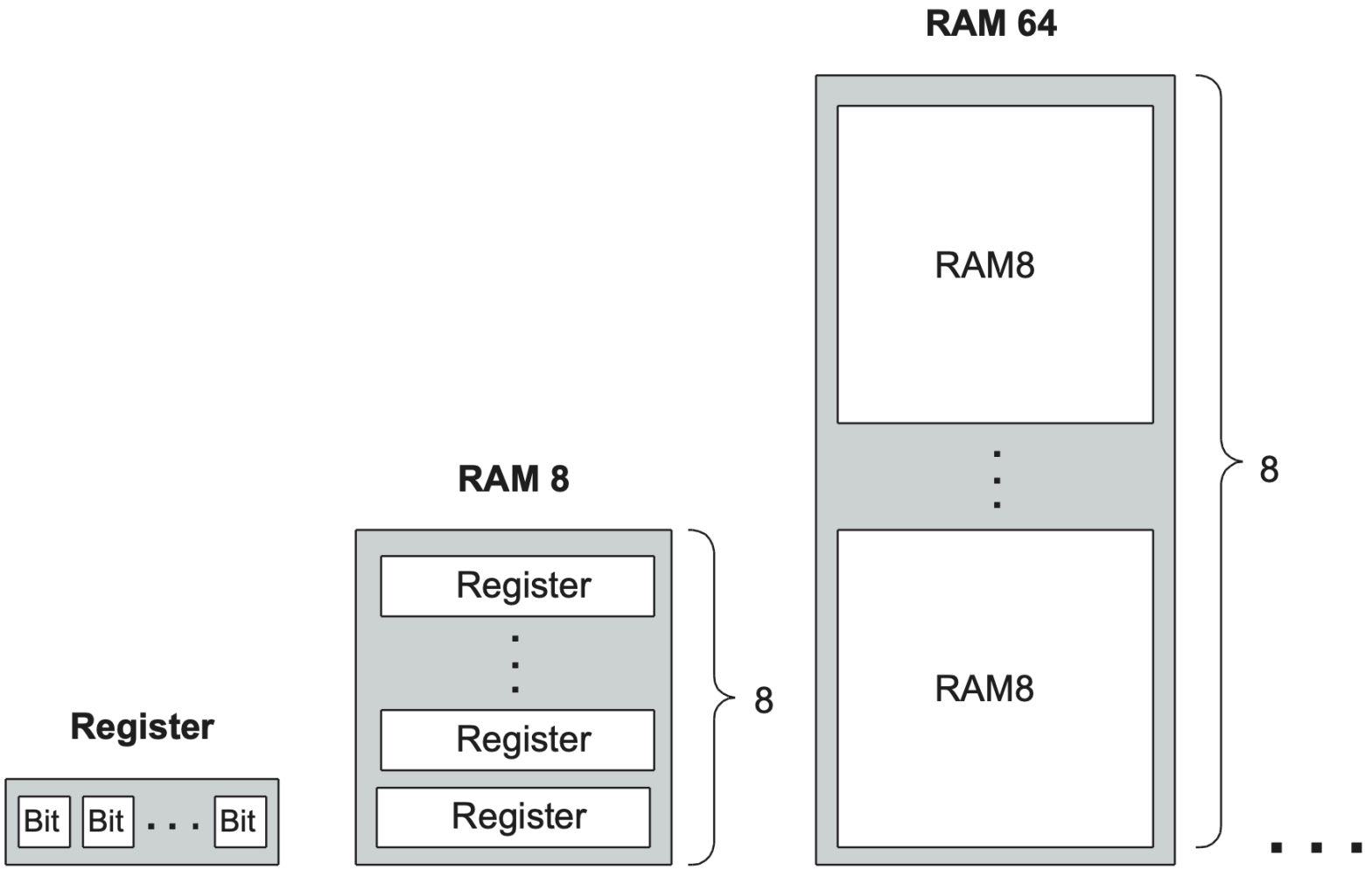
- สร้าง RAM8 โดยใช้
  - 16-bit input /output
  - 3-bit addressing ใช้เ

RAM8.hdl

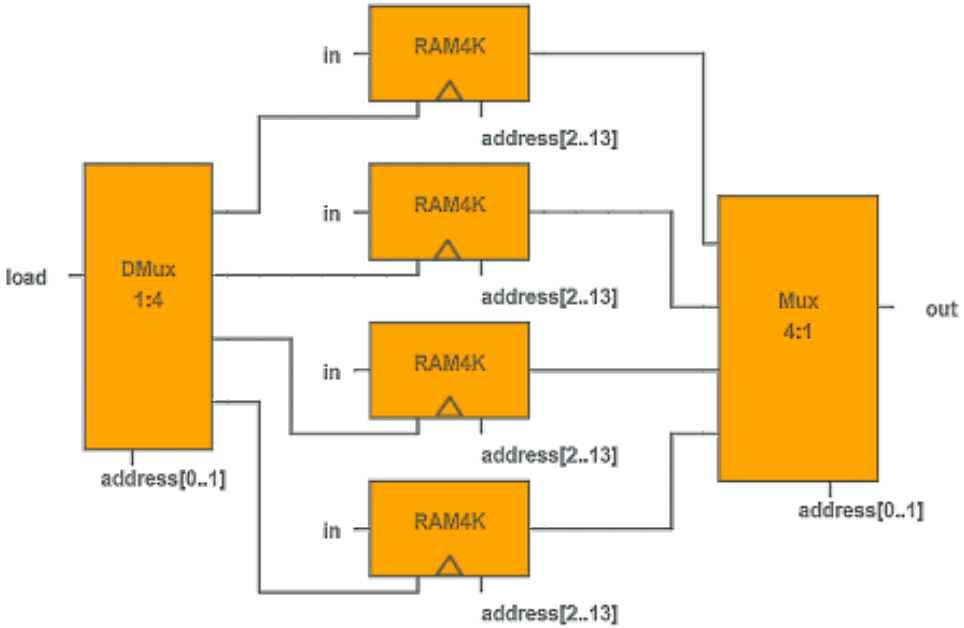
```
/**
 * Memory of 8 registers, each 16 bit-wide. Out holds the value
 * stored at the memory location specified by address. If load=1, then
 * the in value is loaded into the memory location specified by address
 * (the loaded value will be emitted to out after the next time step.)
 */

CHIP RAM8 {
   IN in[16], load, address[3]; OUT out[16];
PARTS:
   DMux8Way(in=load,sel=address,a=a,b=b,c=c,d=d,e=e,f=f,g=g,h=h);
   Register(in=in,load=a,out=oa); Register(in=in,load=b,out=ob);
   Register(in=in,load=c,out=oc); Register(in=in,load=d,out=od);
   Register(in=in,load=e,out=oe); Register(in=in,load=f,out=of);
   Register(in=in,load=g,out=og); Register(in=in,load=h,out=oh);
   Mux8Way16(a=oa,b=ob,c=oc,d=od,e=oe,f=of,g=og,h=oh,sel=address,out=out);
}
```

```
/*
 * Let M stand for the state of the
 * register selected by address.
 * if load(t) then {M=in(t), out(t+1)=M}
 * else                      out(t+1)=M
 */

CHIP RAM8 {
    IN in[16], load, address[3];
    OUT out[16];

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

RAM8, RAM64, … , RAM16K



RAM16K




```
CHIP RAM16K {
   IN in[16], load, address[14];
   OUT out[16];

   PARTS:
   DMux4Way(in=load, sel=address[0..1], a=r0, b=r1, c=r2, d=r3);
   RAM4K(in=in, load=r0, address=address[2..13], out=r0out);
   RAM4K(in=in, load=r1, address=address[2..13], out=r1out);
   RAM4K(in=in, load=r2, address=address[2..13], out=r2out);
   RAM4K(in=in, load=r3, address=address[2..13], out=r3out);
   Mux4Way16(a=r0out, b=r1out, c=r2out, d=r3out,
sel=address[0..1], out=out);
}
```