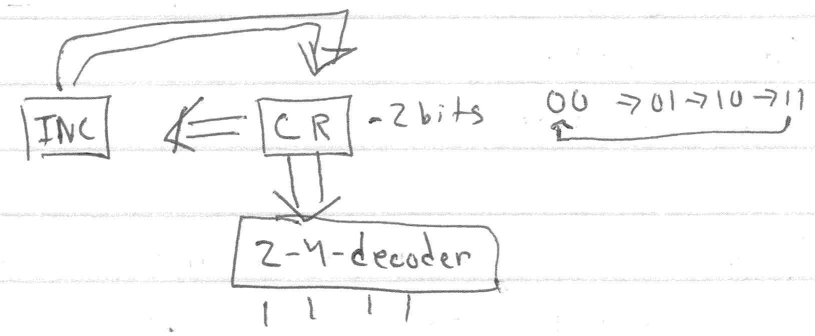
6-2

Clock : N- lines (N=4)

↓ ↓ ↓ ↓
C_0 C_1 C_2 C_3

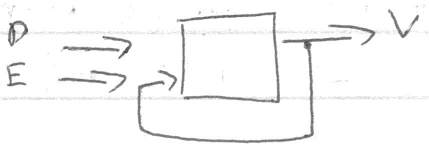


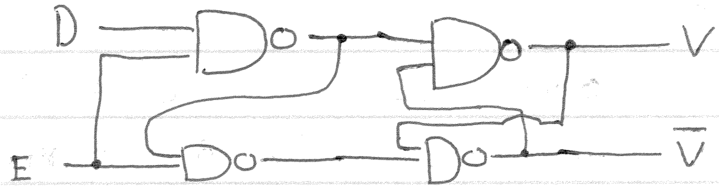
Latch (or 1-bit Register)

IN: Data , Enable

Out : Value

V_{prev}	D	E	V
0 1	0	0	V prev 0 1
V_{prev}	1	0	V prev
–	0	1	0
–	1	1	1





7-1/ 16 registers (5 are given special names for constants:
1, 0, -1, Amask, SMASK)

2 memory buffer registers
Word's are 16-bits MIC-1
Programs are 256 instructions
& instructions are 32-bits

---

3 registers - AC (accumulator) PC (program counter) SP (stack pointer)
4096 memory address (12-bits) → M MAC-1
~~Memory~~ program & data share same space
Instruction/data are 16-bits

LODD (Load Direct) 1 arg (X-addr) ⇒ AC := M[X]

LODD 32 ⇒ AC := M[32]

$\underbrace{0000}$ $\underbrace{xxxx}$ $\underbrace{xxxx}$ $\underbrace{xxxx}$ ⇒ 32 $\boxed{0 \to}$ $\boxed{32}$

TOD (store Direct) 1 arg (X-addr) ⇒ M[X] := AC

$\underbrace{0001}$

ADDD 0010 xxxx → ⇒ AC := AC + M[X]

UBD 0011 x12 ⇒ AC := ~~AC~~ AC - M[X]

POS 0100 x12 ⇒ if AC ≥ 0 then PC := X

ZER 0101 ⇒ if AC == 0 then PC := X

UMP 0110 ⇒ PC := X

JUMP 32 ⇒ 0110 0000 0010 0000 ⇒ 24608

OCO 0111 ⇒ AC := X (0 ≤ x ≤ 4095)

ODL (Load Local) 1000 ⇒ AC := M[SP + X]

LOAD $\underbrace{y}$000 $\underbrace{x12}$ ⇒ AC := M[(if y SP 0) + x]

TOL 1001 ⇒ M[SP+X] := AC

DDL 1010 ⇒ AC := AC + M[SP + x]

BL 1011 ⇒ AC := AC - M[SP+X]

NEG 1100 ⇒ if AC < 0, PC := X

NZE 1101 ⇒ if AC ≠ 0, PC := X

7-2/

CALL 1110 => SP :=SP-1 ; M[SP]:= PC ; PC := X

RETN 1111 1000 0- => PC:= M[SP] ; SP := SP + 1

PUSH 1111 0100 0- => SP:=SP-1 ; M[SP] := AC

POP 1111 0110 0- => AC:=M[SP] ; SP := SP+1

PSHI 1111 0000 0- => SP:=SP-1 ; M[SP] := M[AC]

POPI 1111 0010 0- => M[AC]:= M[SP] ; SP:=BP+1

SWAP 1111 1010 0- => swap AC and SP

INSP 1111 1100 8y => SP := SP + Y (0 ≤ y ≤ 255)

DESP 1111 1110 8y => SP := SP - Y ''

HALT 1111 1111 0- => Stops machine

```
                                   0   0    4095
MAC-1 () initial value of AC, PC, SP      {
  while true {                    FETCH
    I = M[PC]   <-----------------
    switch (I) <- {  <------------ DECODE
                                                  EXECUTE
      case LODD:   AC = M[X] ; continue  )
      ...
      case HALT:  return;
    }
  }
}
```

-1/ Assume that "A" is a location and "B" is a location and "N" is a locati
↳ Ptr to ADATA ↙ N values ↳ Ptr to BDATA ↙ N values ↙ number

Purpose : Push A[i] + B[i] onto the stack for i ∈ [0,N]
Leave AC as the total of all A & B

Purpose : Compute all the Fibonacci numbers the MIC-1 can
↳ Save them to the stack

```
STOL Ø              // AC = 0   =>  m[SP] = Fib(0) = 0
LOCO 1              //          =>  AC = 1
STOL 1              // AC = 1   =>  m[SP+1] = Fib(1) = 1
loop: ADDL Ø        // AC = Fib(n+1)   m[SP] = Fib(n) => AC = Fib(n+2)
      STOL 2        //    m[SP+2] = Fib(n+2)
      INSP 1        // AC = Fib(n'+1)   m[SP] = Fib(n')
                                  n' = n+1
      JPOS loop:
```

Instruction encoding RISC / MIPS