Announcements

- HW #6 was handed out
- Last day for MT #1 regrades is 10/31

Completeness of Instructions

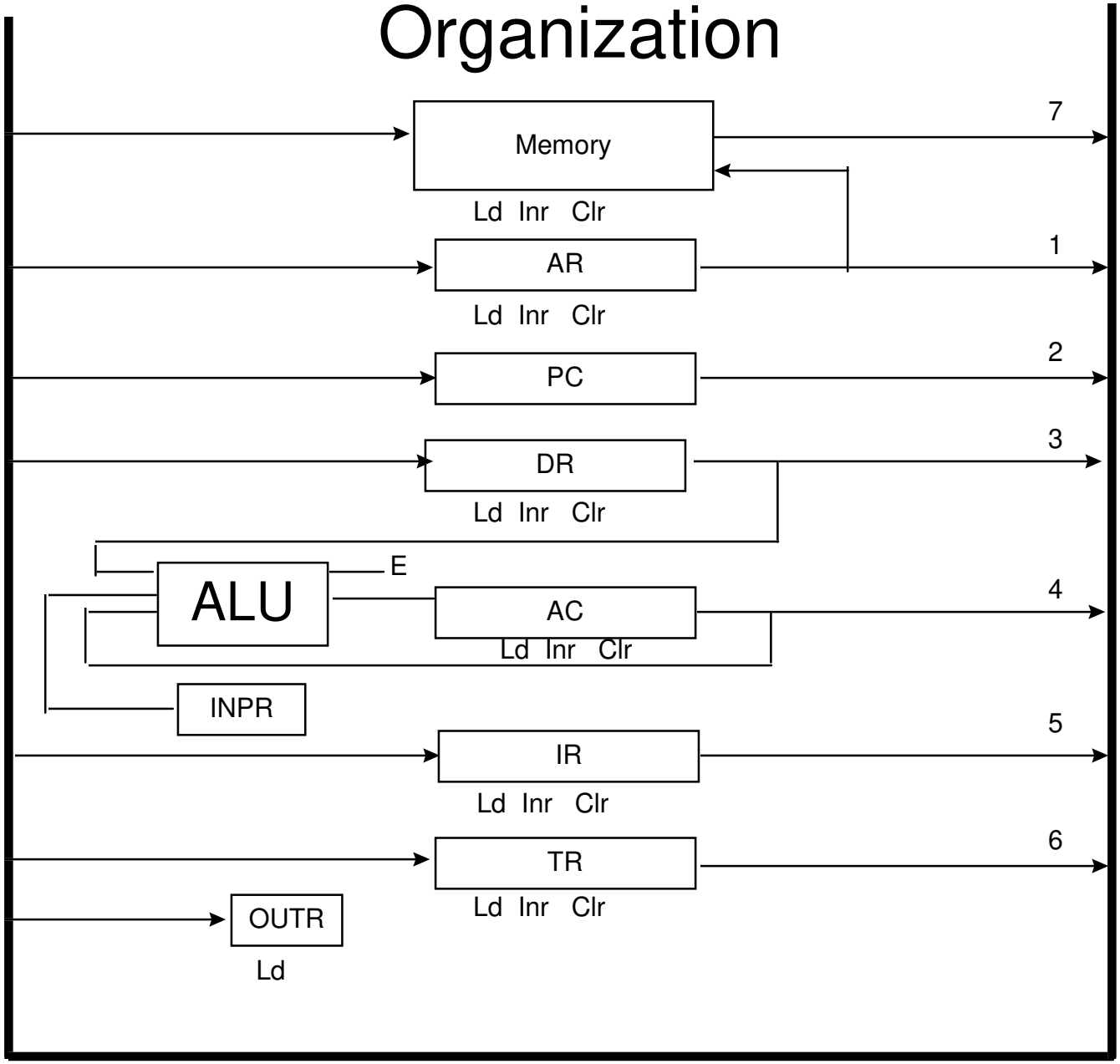
- Need to ensure that the computer can execute any computable function.
  - equivalent to a turning machine
  - more instructions make the machine faster, but not more powerful
- Arithmetic
  - add, complement, increment
  - AND and NOT (provides NAND which is enough)
- Data movement instructions
  - LDA, STA
- Control instructions
  - check status of words/AC
  - provide a way to change the next instruction to execute
- Input/Output Instructions

Steps Involved in Executing an Instruction

```
while (not done) {
    Read the instruction from memory
    figure out what the instruction is (called decode)
    read the effective address from memory
        # only required if the instruction is indirect
    execute the appropriate micro-instructions
}
```

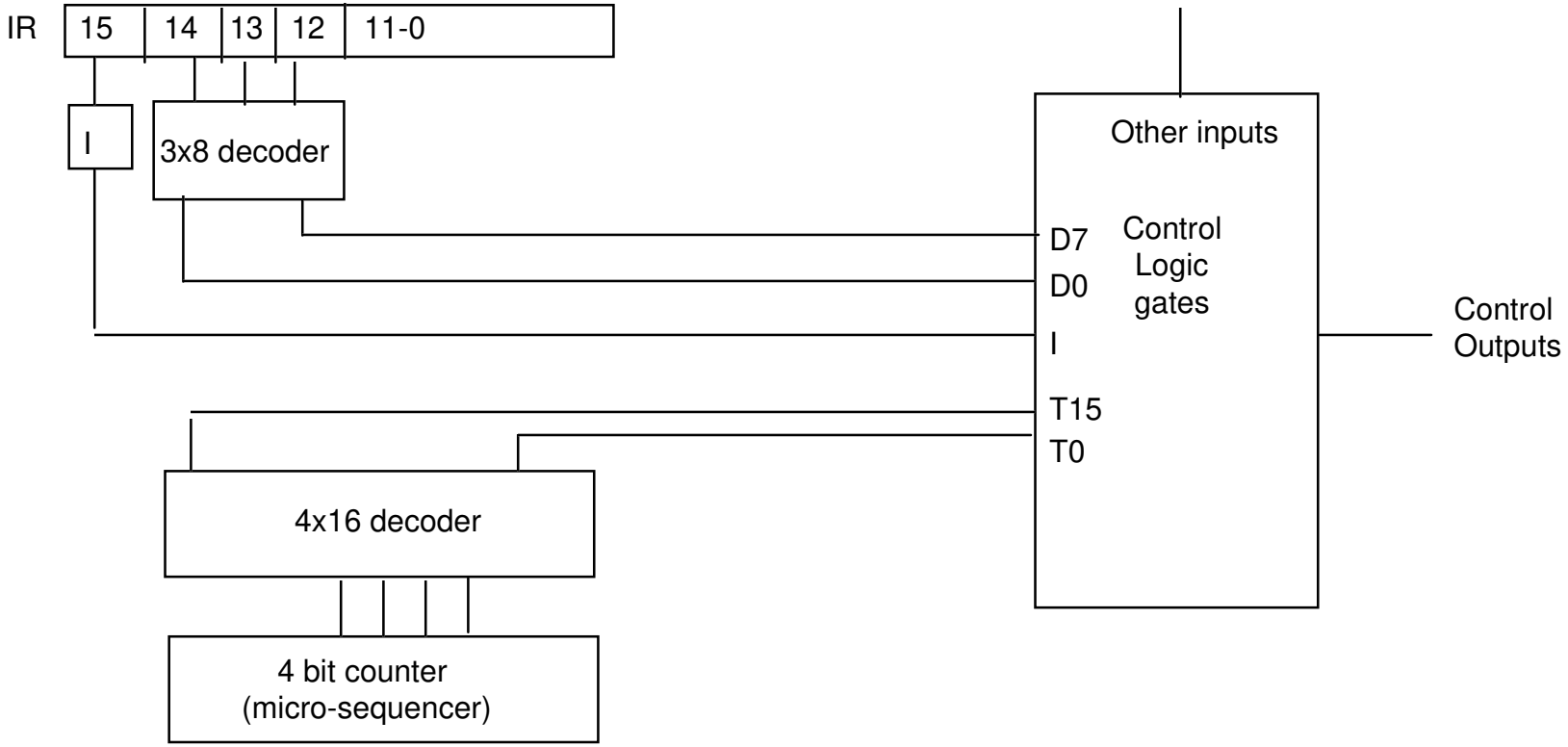
- Also need to:
  - update the PC
  - specify the correct control lines as output
    - bus, registers, memory, ALU

Organization



Micro-Sequencer Control

- To keep track of steps in executing an instruction
  - use a micro-sequencer
  - really just a fancy name for a counter with clear



Micro-sequencer Steps for fetch/decode

- The state of the micro-sequencers tells us which step in executing an instruction we are at
- The Steps involved are:

  T0: AR <- PC

  T1: IR <- M[AR], PC <- PC + 1

  T2: D0,….D7 <- decode(IR(12-14)),

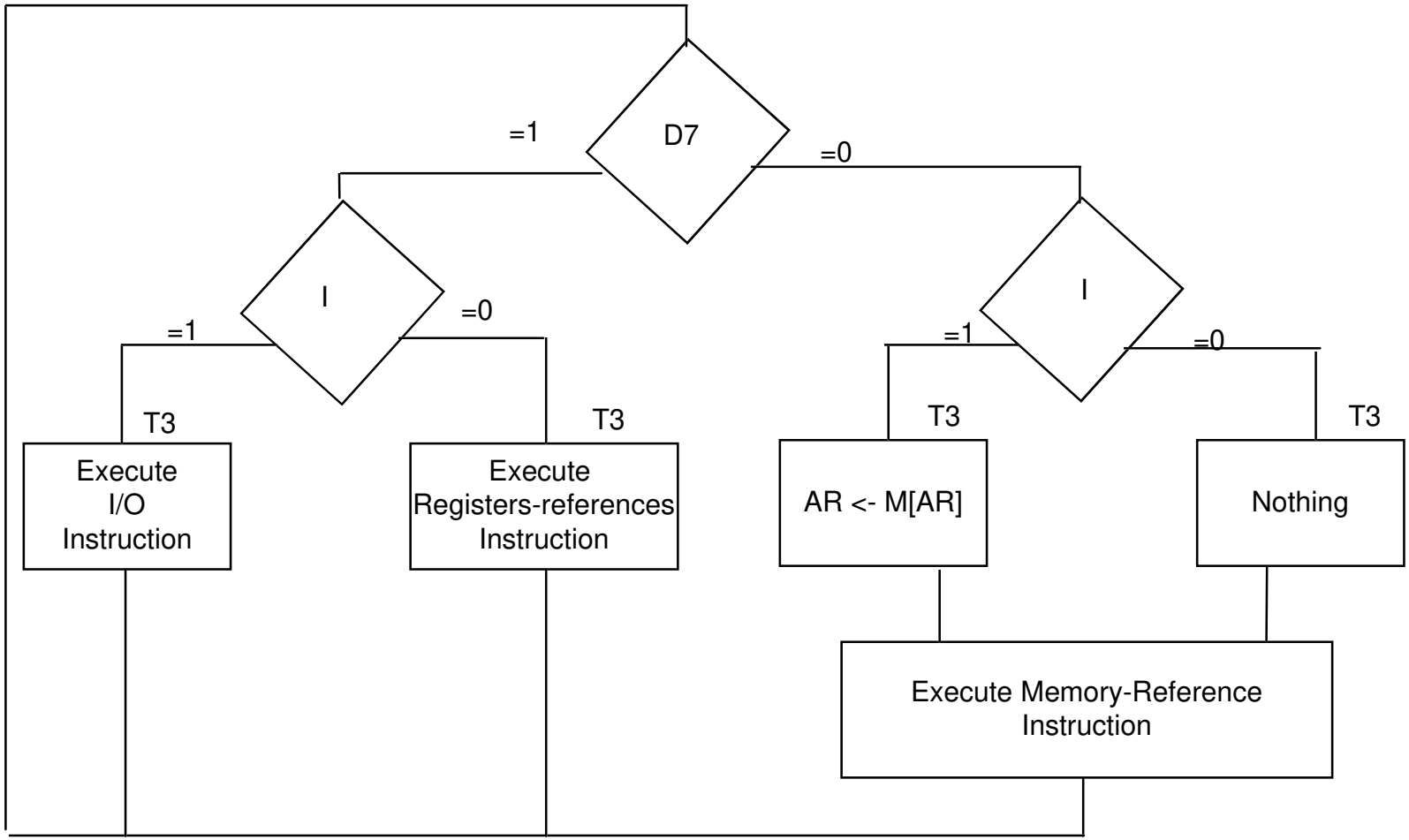
  AR <- IR (0-11), I <- IR(15)
- At T0:
  - PC on bus so S2=0, S1=1, S0=0
  - $LD_{AR} = 1$

Fetch and Decode (cont)

- At T1:
  - S2 = S1 = S0 = 1
  - read line on memory = 1
  - $LD_{IR} = 1$
  - $INR_{PC} = 1$
- At T2:
  - S2=S0=1, S1=0
  - $LD_{AR} = 1$
  - $LD_{I} = 1$

What Happens after Decode?

Answer: It depends on the instruction



Executing the Instruction

- Use I, D7, and T3 to control which micro-instruction gets executed
  - D7' I T3: AR <- M[Ar]
  - D7' I' T3: nothing
  - D7 I' T3: execute register insn
  - D7 I T3: execute I/O instruction

Executing Register Instructions

- D7 I' T3 = r (common to all reg instructions)
- IR(I) = Bi (ith bit of instruction)

	R:	SC <- 0	
CLA	rB11:	AC <- 0	Clear AC
CLE	rB10:	E <- 0	Clear E
CMA	rB9:	AC <= AC'	Comp AC
CME	rB8:	E <- E'	Comp E
CIR	rB7:	AC <- shr AC, AC(15)<- E, E <- AC(0)	circulate right
CIL	rB6:	AC <- shl AC, AC(0)<- E, E <- AC(15)	circulate left
INC	rB5:	AC <- AC + 1	increment AC
SPA	rB4:	if AC(15) = 0 PC <- PC+ 1	skip if positive
SNA	rB3:	if AC(15) = 1 PC<-PC +1	skip if negative
SZA	rB2:	if AC = 0 PC <- PC + 1	skip if zero
SZE	rB1:	if E=0 PC <- PC + 1	skip if E zero
HLT	rB0:	S <- 0 (start/stp FF)	halt computer

Memory Reference Instructions

- Memory instructions may take more than one clock

AND	D0T4	DR <- M[AR]
	D0T5	AC <- AC ^ DR, SC <- 0
ADD	D1T4	DR <- M[AR]
	D1T5	AC <- AC ^ DR, SC <- 0
LDA	D2T4	DR <- M[AR]
	D2T5	AC <- DR, SC <- 0
STA	D3T4	M[AR] <- AC, SC <- 0
BUN	D4T4	PC <- AR, SC <- 0
BSA	D5T4	M[AR] <- PC, AR <- AR + 1
	D5T5	PC <- AR, SC <- 0
ISZ	D6T4	DR- M[AR]
	D6T5	DR <- DR + 1
	D6T6	M[AR] <- DR, if (DR=0) PC <- PC + 1, SC <- 0