SCRAM Instructions II

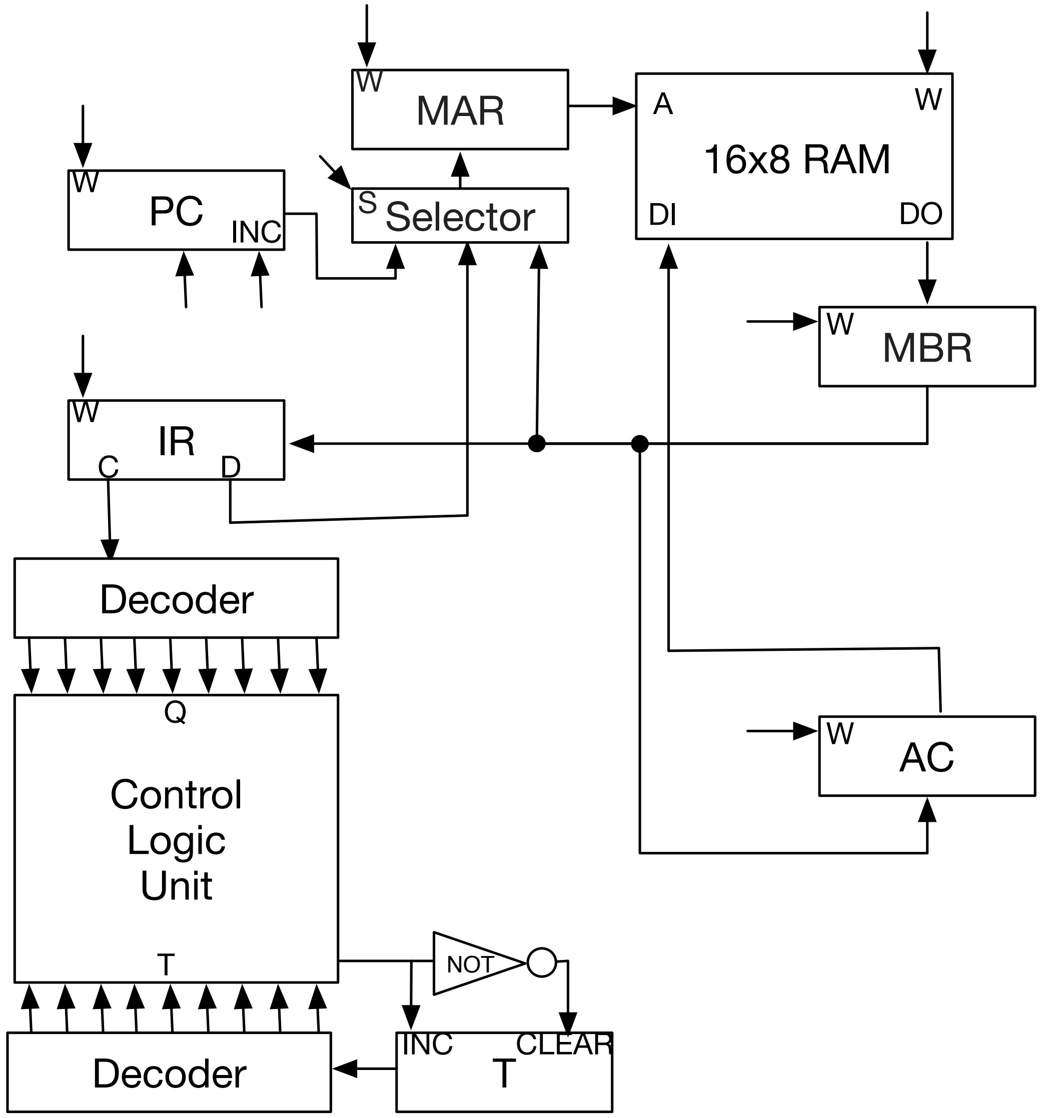
Philipp Koehn

23 February 2018

Reminder

- Fully work through a computer
  - circuit
  - assembly code
- Simple but Complete Random Access Machine (SCRAM)
  - every instruction is 8 bit
  - 4 bit for op-code: 9 different operations (of 16 possible)
  - 4 bit for address: 16 bytes of memory
- Background reading on web page
  - The Random Access Machine
  - The SCRAM

Circuit (At This Point)



Instruction Fetch

- Retrieve instruction from memory
- Increase program counter

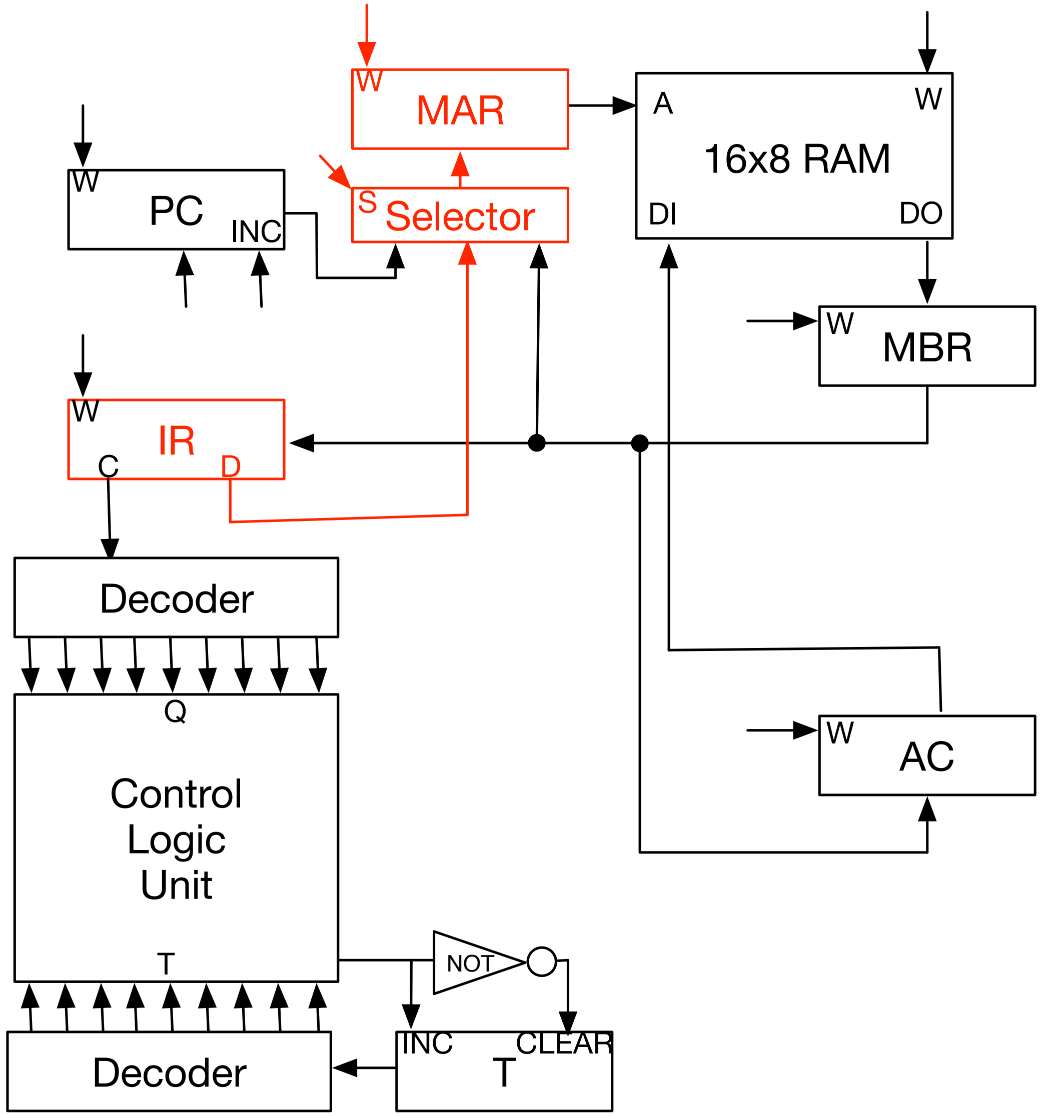
Time	**Command**
t_0	MAR $\leftarrow$ PC
t_1	MBR $\leftarrow$ M, PC $\leftarrow$ PC + 1
t_2	IR $\leftarrow$ MBR

Micro Program for STA

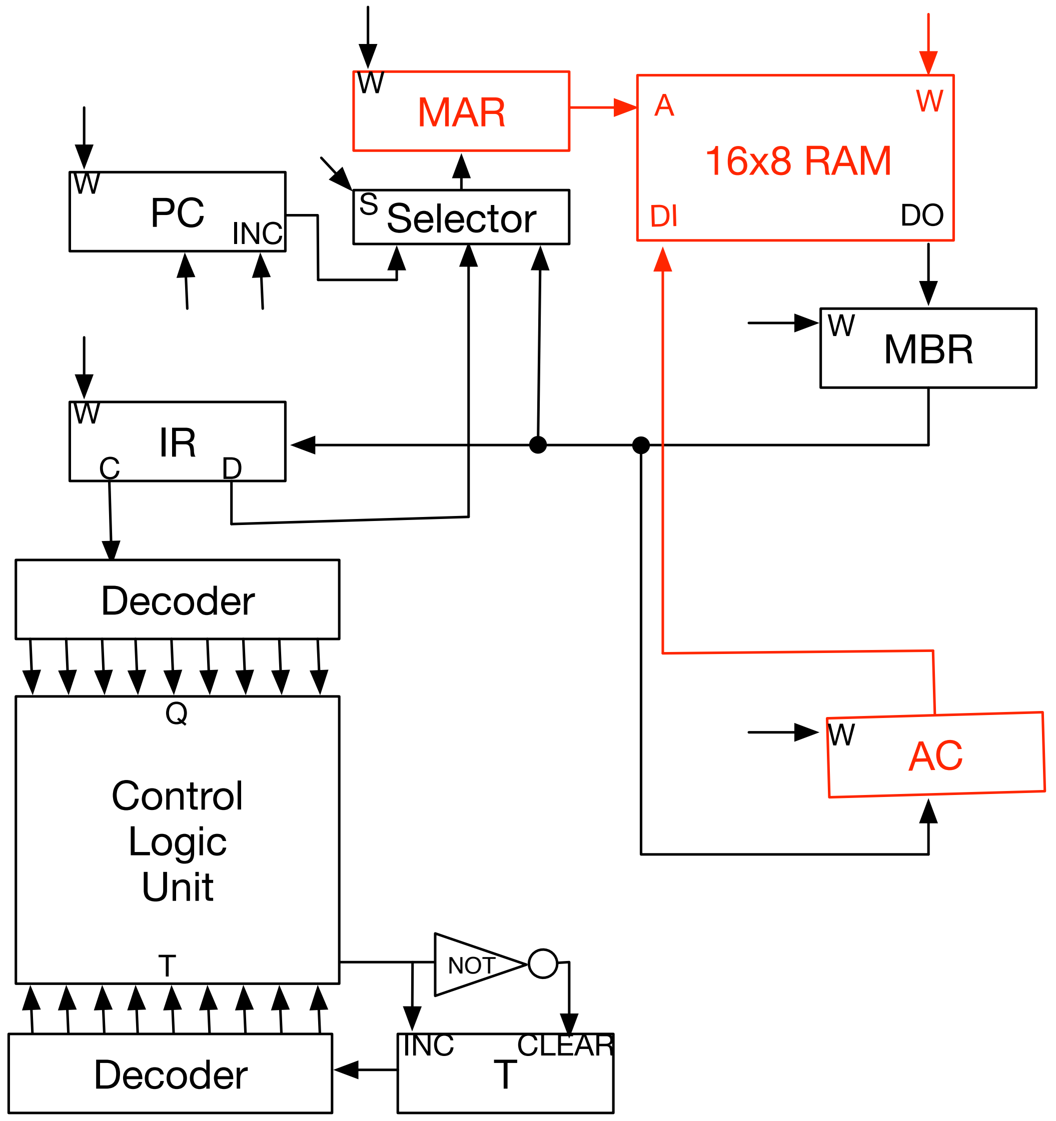
- Store value from accumulator

Op Code	**Time**	**Command**
q_3	t_3	MAR $\leftarrow$ IR(D)
q_3	t_4	M $\leftarrow$ AC

q_3 t_3: MAR $\leftarrow$ IR(D)

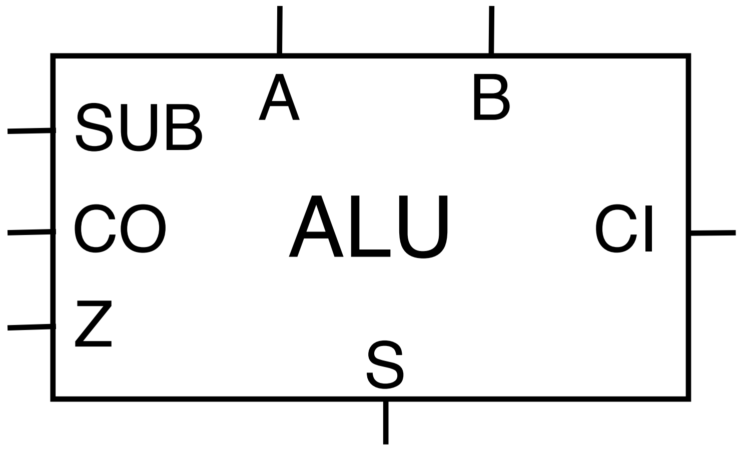


q_3 t_4: M $\leftarrow$ AC



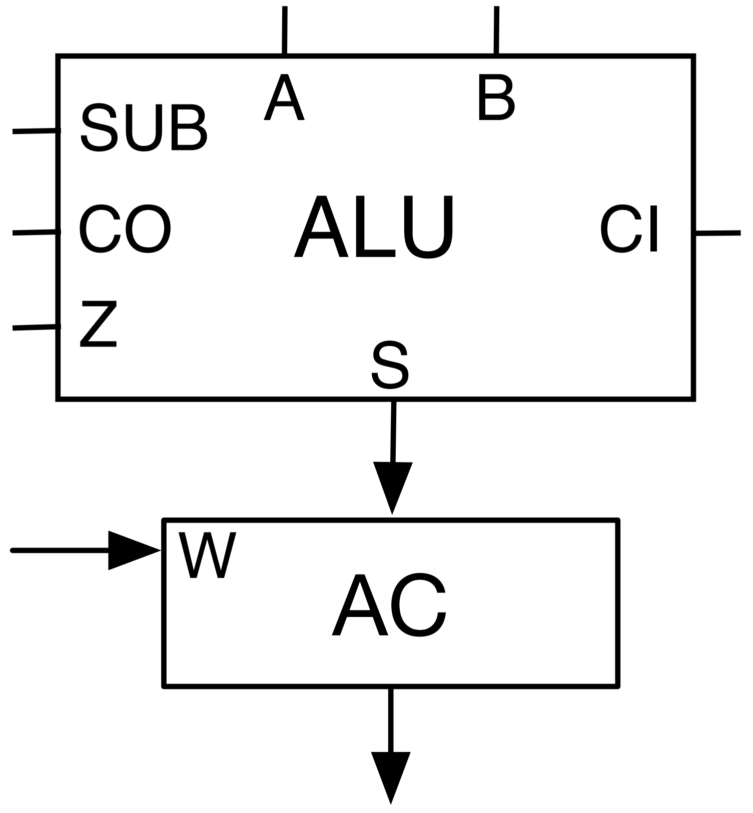
arithmetic logic unit

Arithmetic Logic Unit



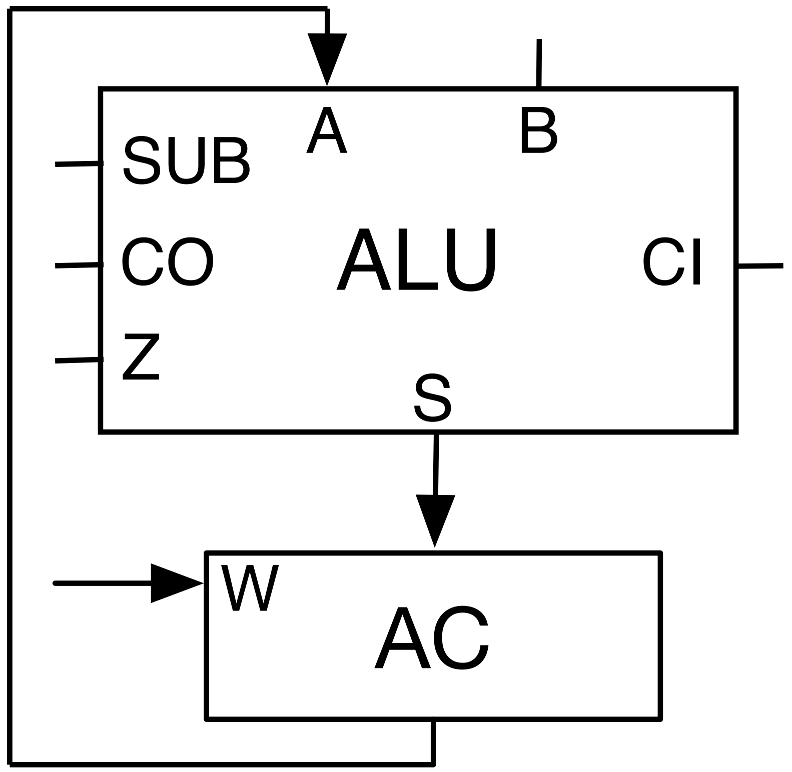
- Adds two numbers: S=A+B
- With subtraction flag: S=A-B
- Overflow handling with carry in (CI) and carry out (CO)
- Zero flag: set if result of operation is 0

Accumulator



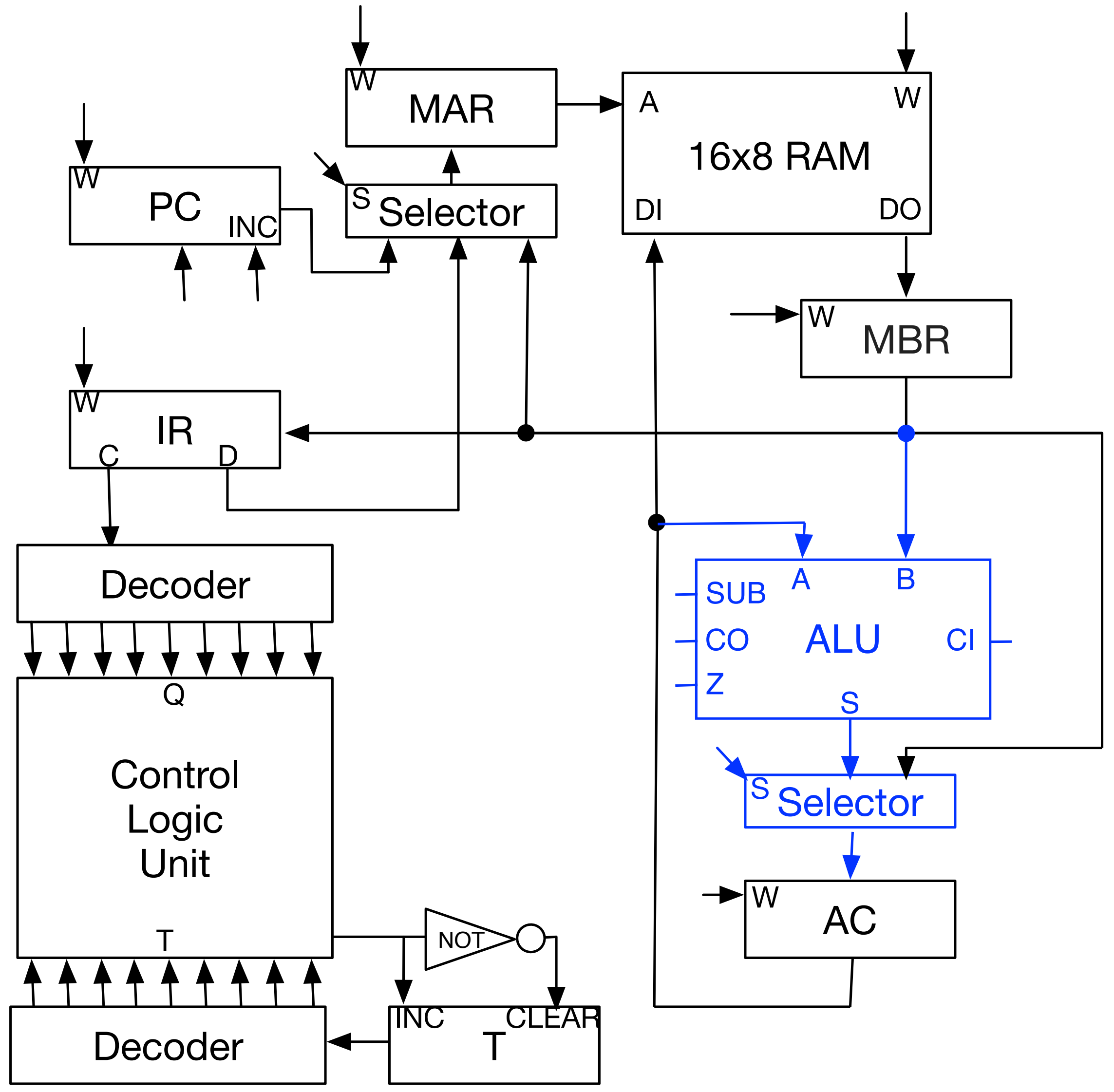
- Store result of ALU operation in accumulator (AC)

AC = AC $\pm$ B



- Accumulator feeds back into ALU
- Operations are AC = AC + B or AC = AC - B

ALU in Circuit



add

ADD: Add to Accumulator

- Add value from memory address to accumulator

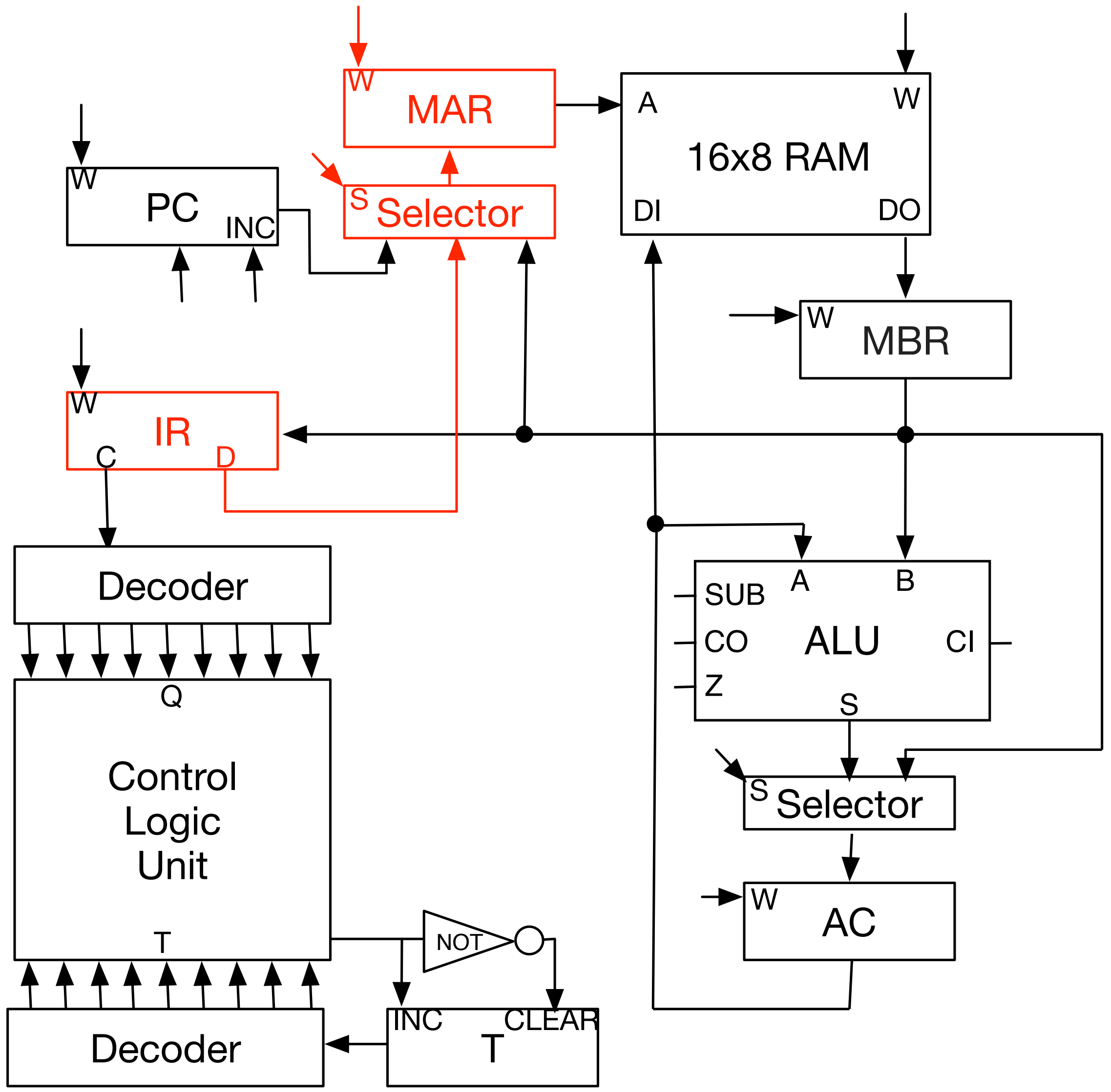
- Steps
  - load value of specified memory address
  - use that value as a memory address (second lookup)
  - store value from second lookup into accumulator

Micro Program for ADD

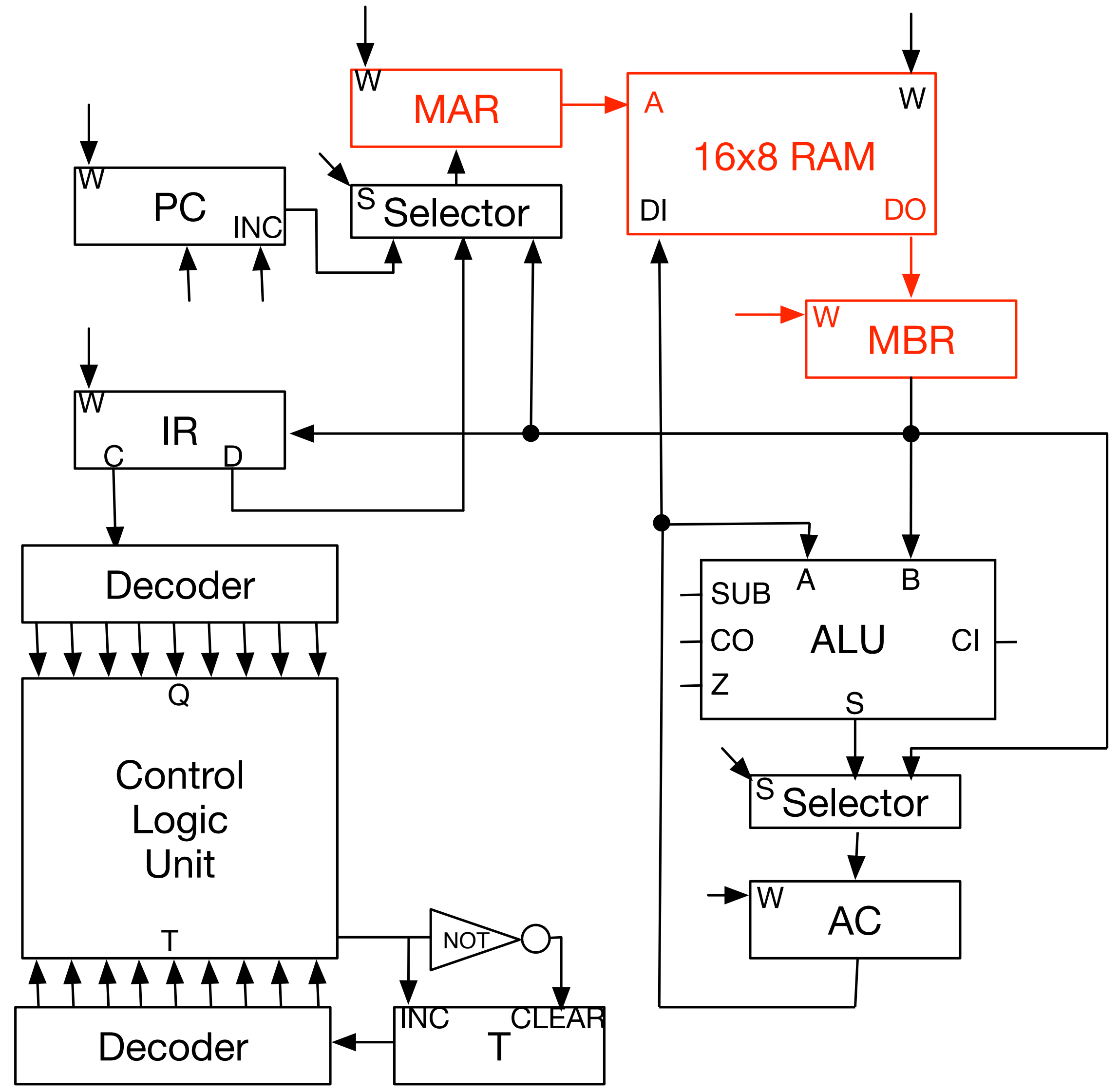
- Load indirectly into accumulator

Op Code	**Time**	**Command**
q_5	t_3	MAR $\leftarrow$ IR(D)
q_5	t_4	MBR $\leftarrow$ M
q_5	t_5	AC $\leftarrow$ AC + MBR

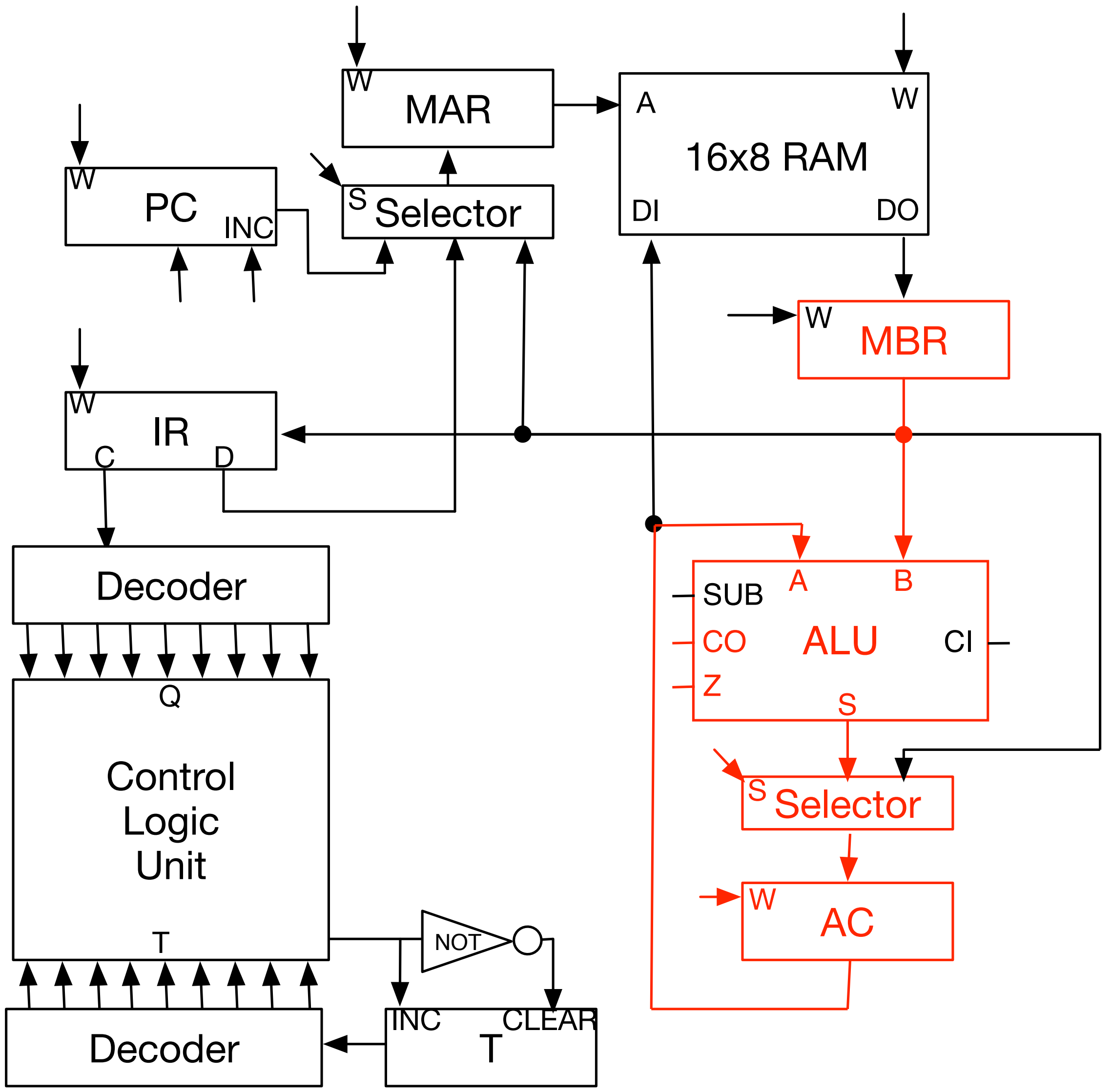
q_5 t_3: MAR $\leftarrow$ IR(D)



q_5 t_4: MBR $\leftarrow$ M



q_5 t_5: AC $\leftarrow$ AC + MBR



sub

SUB: Subtract from Accumulator

- Subtract from accumulator the value from memory

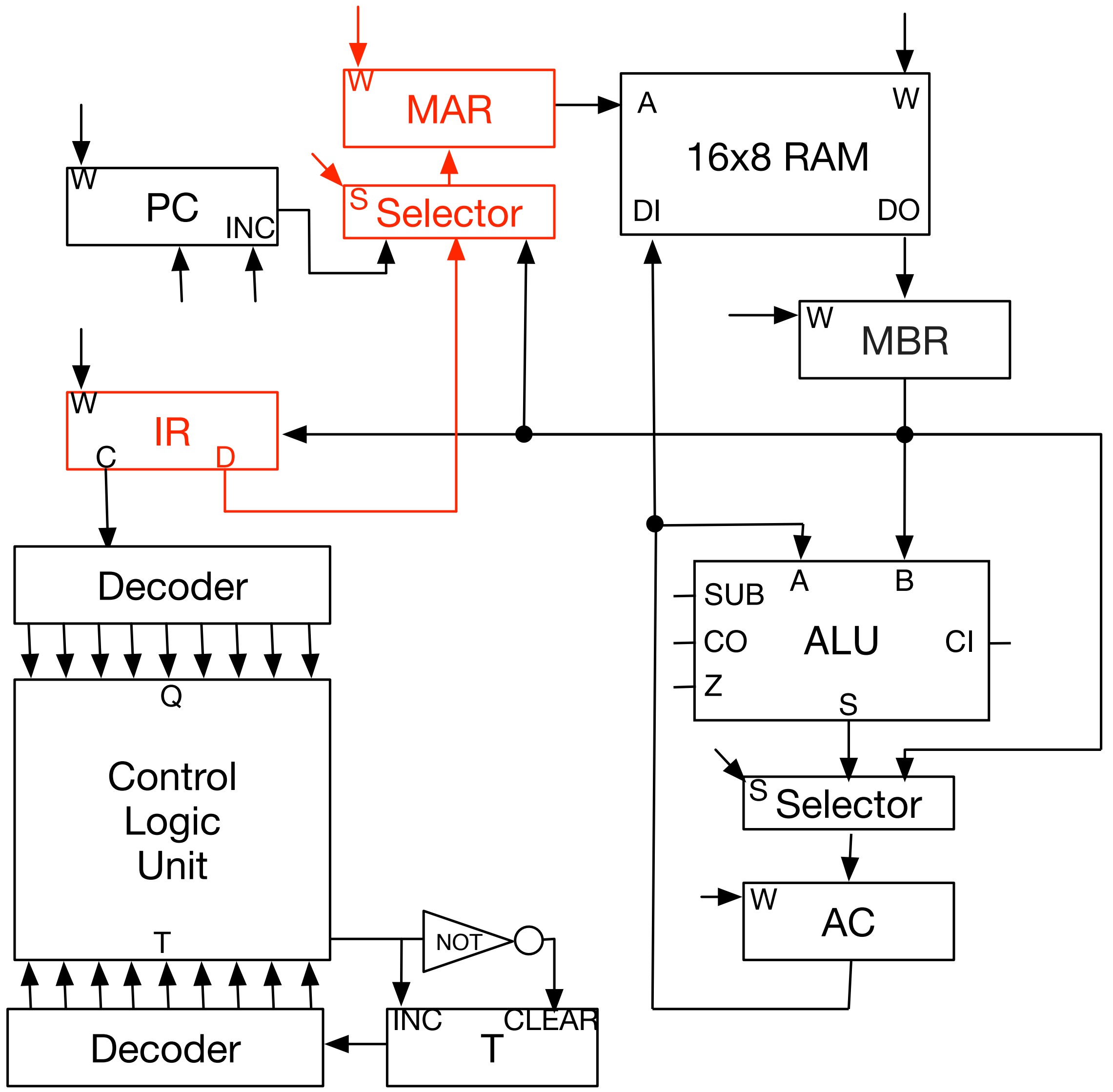
- Same as ADD, just set subtraction flag of ALU

Micro Program for SUB

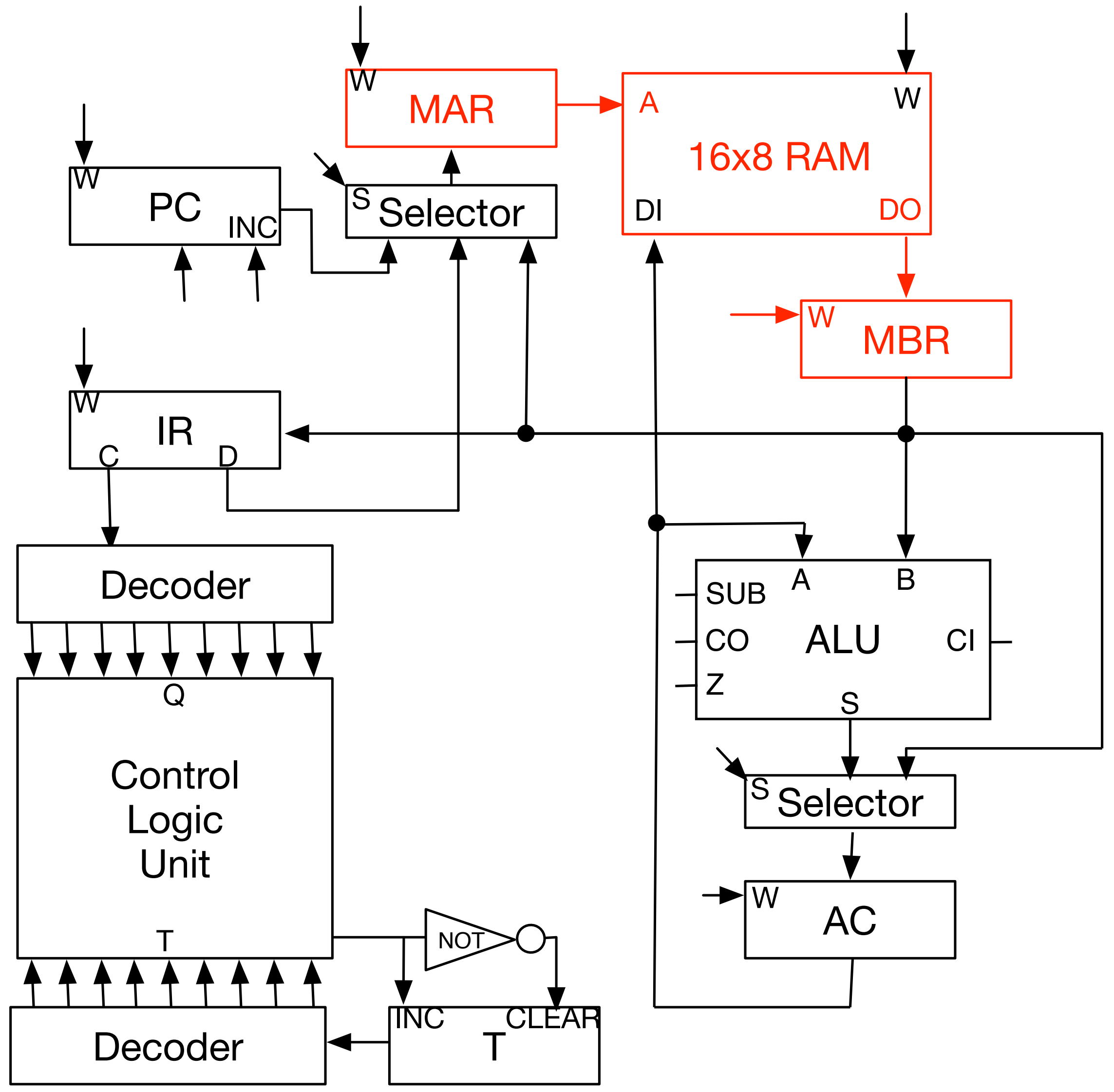
- Load indirectly into accumulator

Op Code	**Time**	**Command**
q_5	t_3	MAR $\leftarrow$ IR(D)
q_5	t_4	MBR $\leftarrow$ M
q_5	t_5	AC $\leftarrow$ AC - MBR

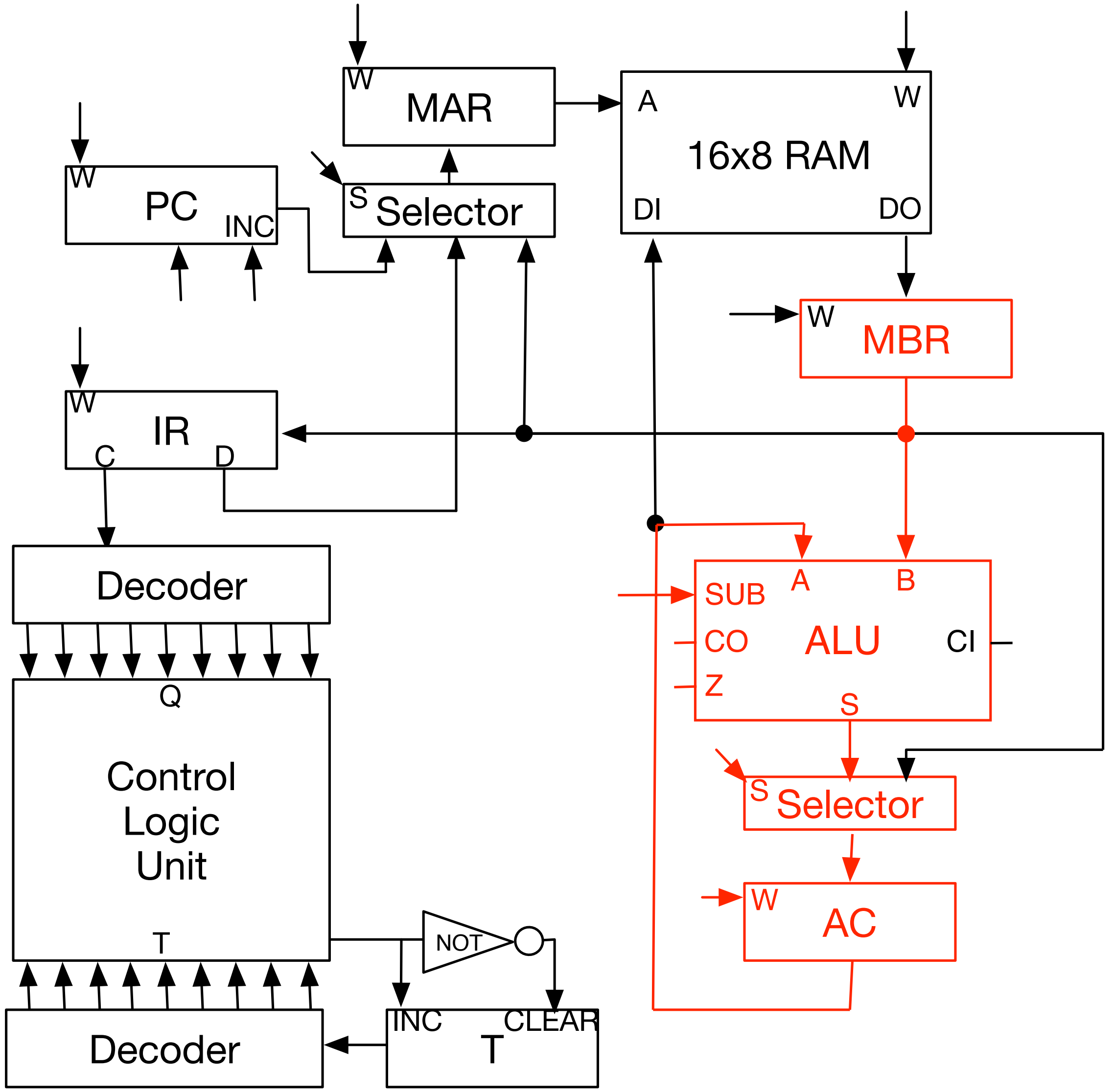
q_5 t_3: MAR $\leftarrow$ IR(D)



q_5 t_4: MBR $\leftarrow$ M



q_5 t_5: AC $\leftarrow$ AC + MBR



`jmp`

Program Counter (PC)

- Position of the next instruction is stored in program counter
- This gets updated during instruction fetch

	Time	Command
	t_0	MAR $\leftarrow$ PC
	t_1	MBR $\leftarrow$ M
	t_2	IR $\leftarrow$ MBR
$\Rightarrow$	t_3	PC $\leftarrow$ PC + 1

JMP: Jump

- Assign value to position of the next instruction

- Sequencing of micro program
  - instruction fetch (includes program counter inc)
  - command-specific micro instructions

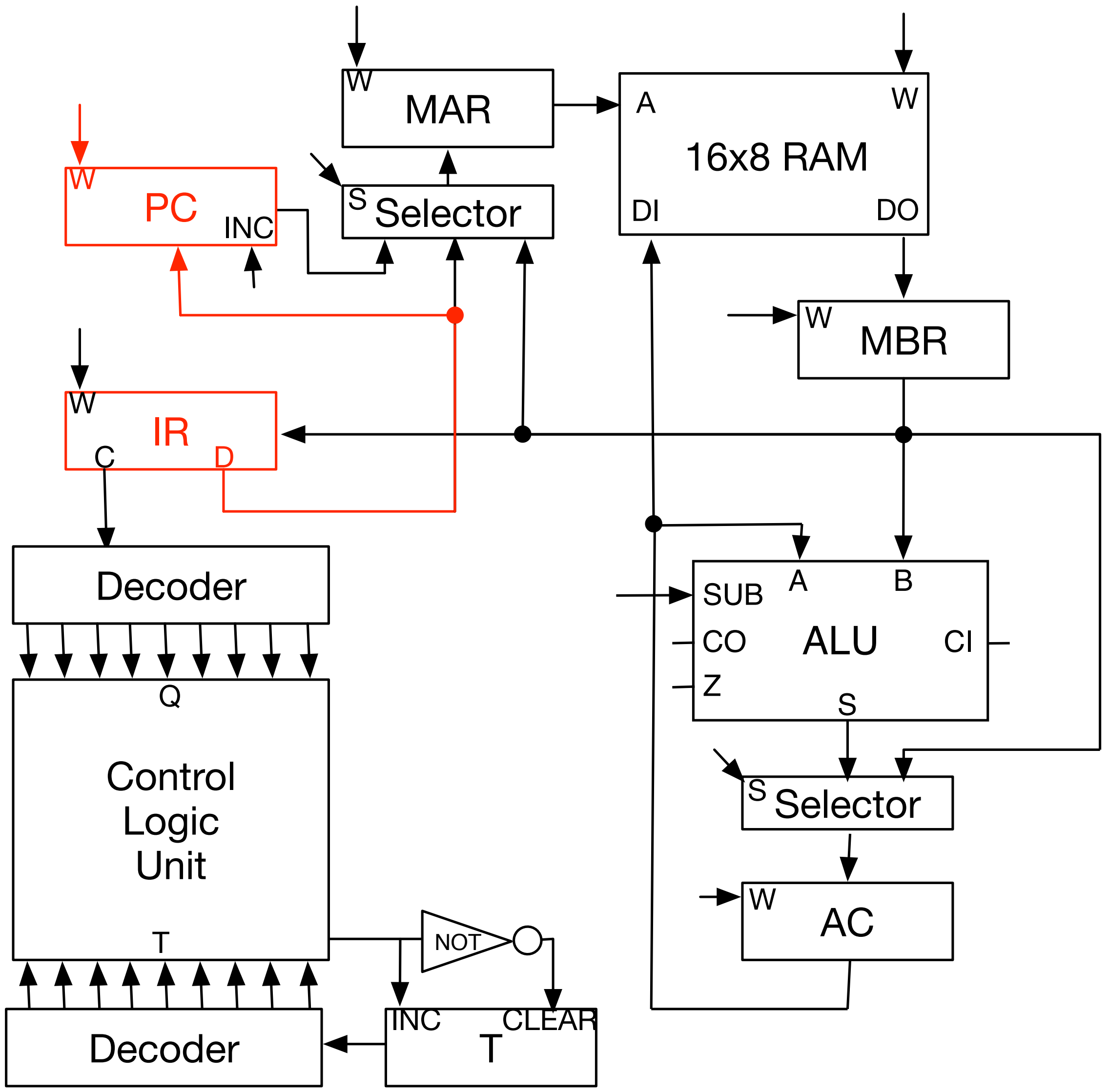
- No problem that program counter gets modified twice

Micro Program for JMP

- Change program counter to specified address

Op Code	**Time**	**Command**
q_7	t_3	PC $\leftarrow$ IR(D)

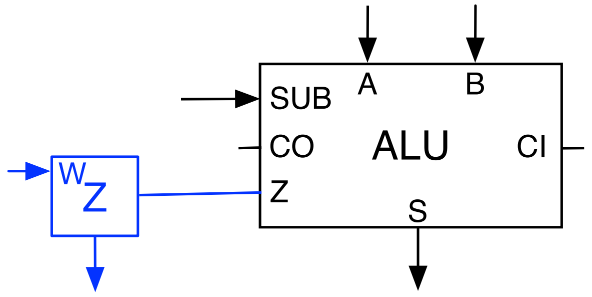
q_7 t_3: PC $\leftarrow$ IR(D)



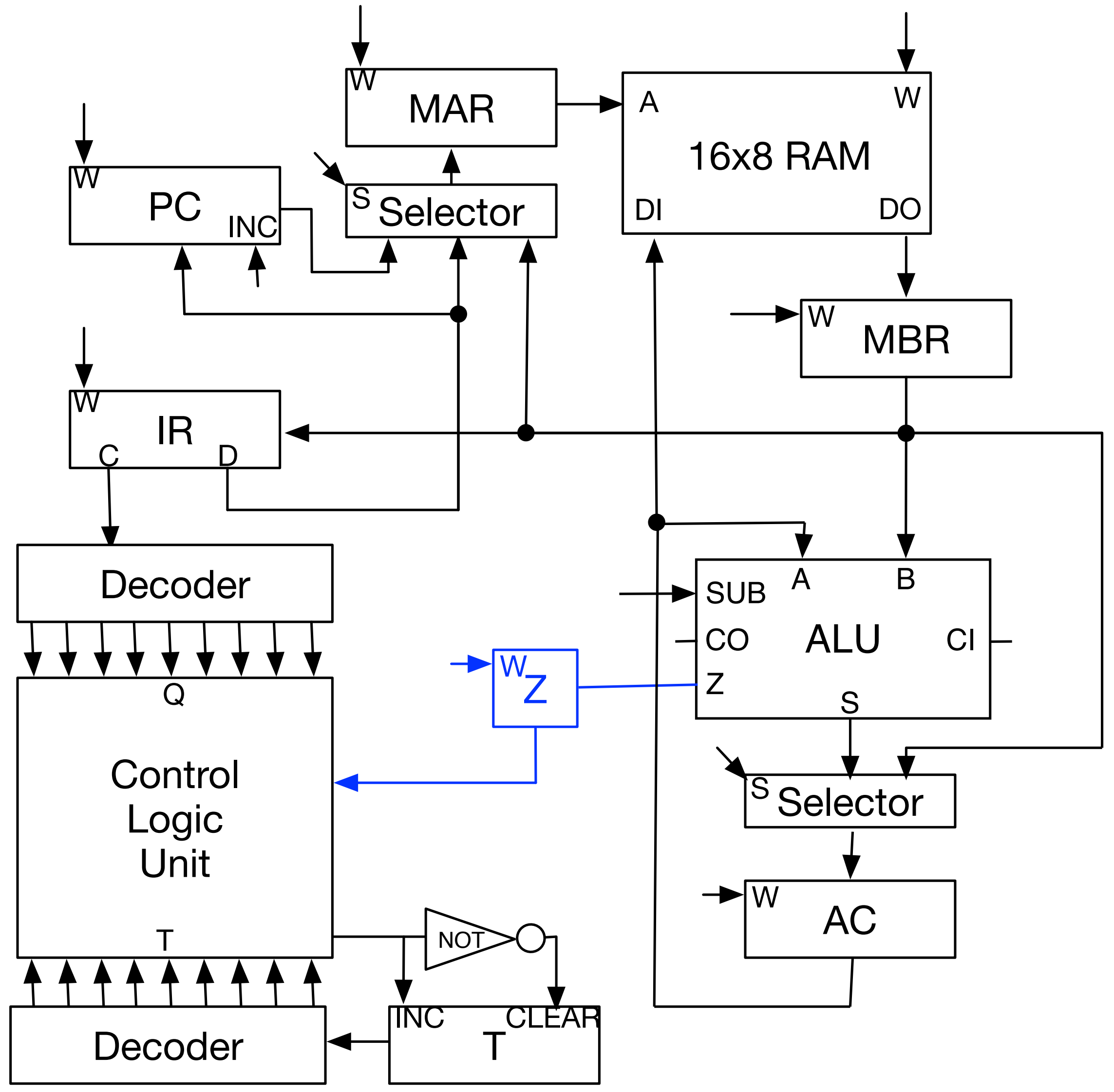
`jpz`

Zero Flag

- Zero flag
  - set when result of a ALU operation is 0
  - stored in flag



Z Flag in Circuit



Micro Program for JPZ

- Z flag is a condition for executing a micro program (same as JMP)

Zero	**Op Code**	**Time**	**Command**
1	q_7	t_3	PC $\leftarrow$ IR(D)

- If not set, no micro program is executed