SCRAM Introduction

Philipp Koehn

13 September 2019

A Simple Computer

- 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 SCRAM"

Operations

Operation	Code	Description
HLT	0000	Halt, stop execution
LDA	0001	Load value from memory into accumulator
LDI	0010	Indirectly load value from memory into accumulator
STA	0011	Store value from accumulator into memory
STI	0100	Indirectly store value from accumulator into memory
ADD	0101	Add value from memory to accumulator
SUB	0110	Subtract value from memory from accumulator
JMP	0111	Jump to specified address
JPZ	1000	Jump to specified address if zero flag is set

Indirect Load?

- LDA x
  - loads the value from address x
- LDI x
  - looks up the value at address x
  - treats that value as an address
  - loads the value at that address
- Indirect load = use of pointer variable

Instruction Encoding

Instruction	**Data**
0 0 0 1	0 1 0 1
LDA	5

Load contents from address 5 into memory

A Simple Program

Address	Code		Meaning	
	op-code	data	operation	data
0	0001	0100	LDA	4
1	0101	0101	ADD	5
2	0011	0100	STA	4
3	0111	0000	JMP	0
4	1111	0000	DAT	0
5	1111	0001	DAT	1

- Note: DAT is not a real instruction
- Produces sequence of numbers:

  0, 1, 2, 3, 4, ..., 255, 0, ...

components

Memory

- 4 bits to address memory

$\Rightarrow$ 16 different values

$\Rightarrow$ 16 byte address space

- We need to build circuitry to retrieve and store values

Registers

- Accumulator (AC)
  - can be directly accessed from logic units
  - used to store the results of computations
- Program counter (PC)
  - memory address of current instruction
- Instruction register (IR)
  - contains current instruction
  - breaks it down into operation code
- Memory registers
  - memory access register (MAR): address to retrieve value
  - memory buffer register (MBR): retrieved value

Arithmetic Logic Unit (ALU)

- Can do addition and subtraction
- Operands
  - operand 1: accumulator
  - operand 2: adress specified in instruction
  - result: accumulator
- Zero flag: result of operation is zero
- Carry flag: operation results in overflow / underflow

Program Counter

- 4 bit current memory address of instruction
- Typically increased by 1 during each instruction execution
- Can also be changed by jump instructions (JMP, JPZ)

Control Logic Unit

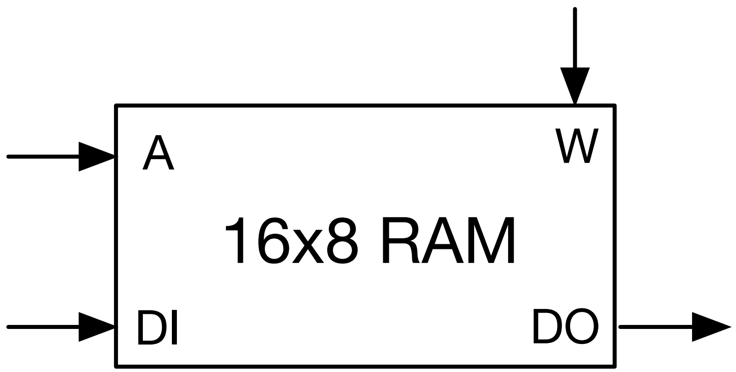
- Decodes the op code
- Selects instruction logic
- Instruction logic: microprogram
  (sequence of register transfers)
- More detail on that in a bit...

Putting it All Together

```
+---------------------------------+---------------------------------+
|                                 |                                 |
|                                 |              Memory             |
|       Program Counter (PC)      |                                 |
|                                 |   Memory access register (MAR)  |
|    Instruction Register (IR)    |                                 |
|                                 |   Memory buffer register (MBR)  |
|                                 |                                 |
+---------------------------------+---------------------------------+
|                                 |                                 |
|                                 |                                 |
|                                 |          Accumulator (A)        |
|    Control Logic Unit (CLU)     |                                 |
|                                 |   Arithmetic Logic Unit (ALU)   |
|                                 |                                 |
|                                 |                                 |
+---------------------------------+---------------------------------+
```

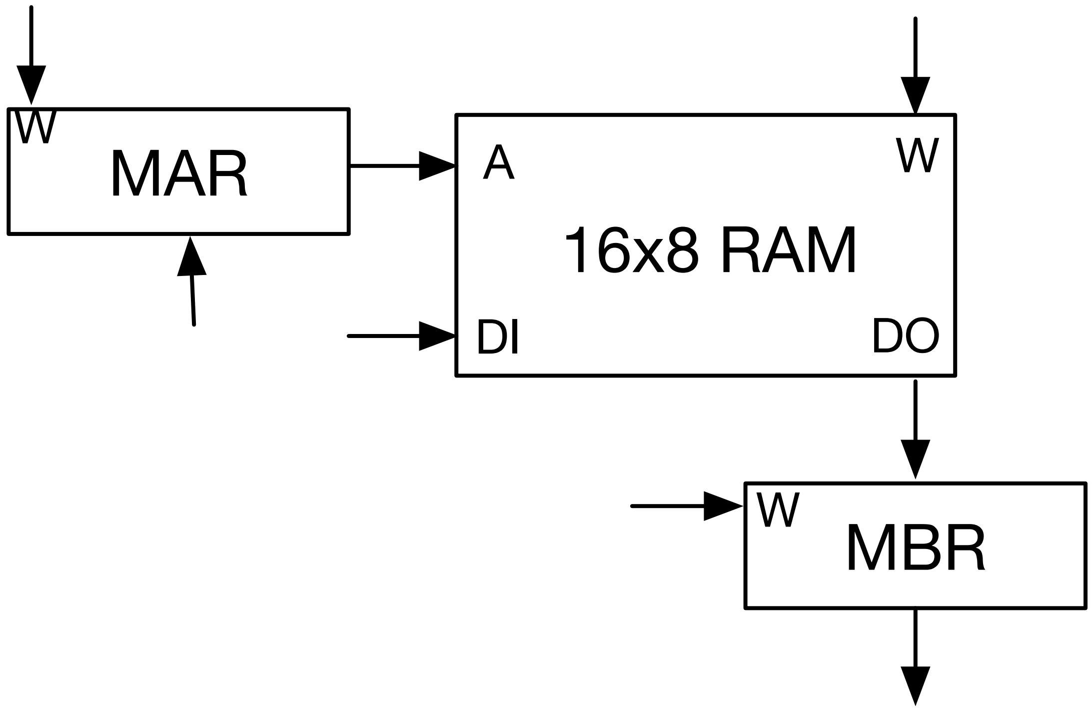
memory

16x8 Bit RAM



- 16 byte of memory
- Inputs
  - address (A)
  - data in (DI)
  - write flag (W)
- Output
  - data out (DO)

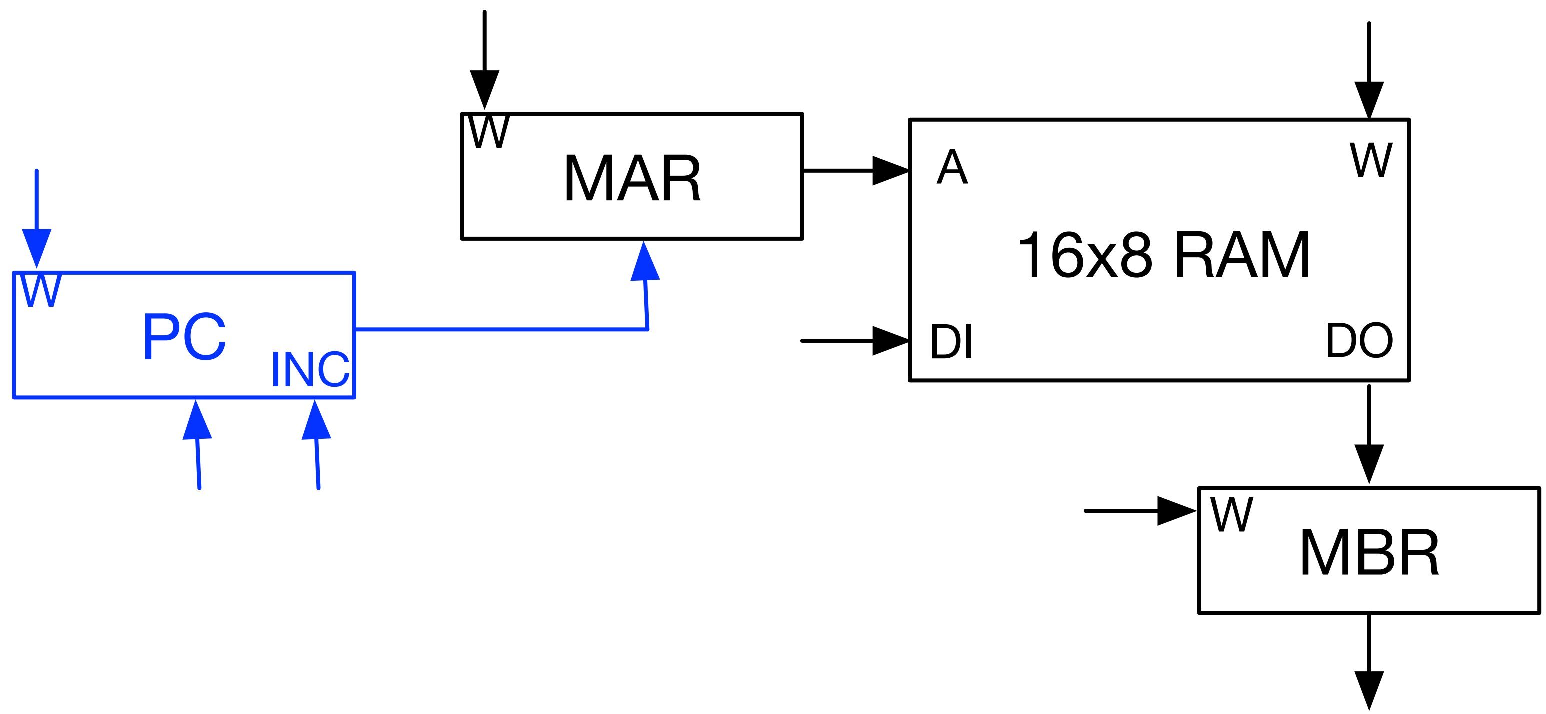
Memory with its Registers



- Memory address register (MAR)
- Memory buffer register (MBR)
- Each of them has a write flag (there will be a few more of them...)

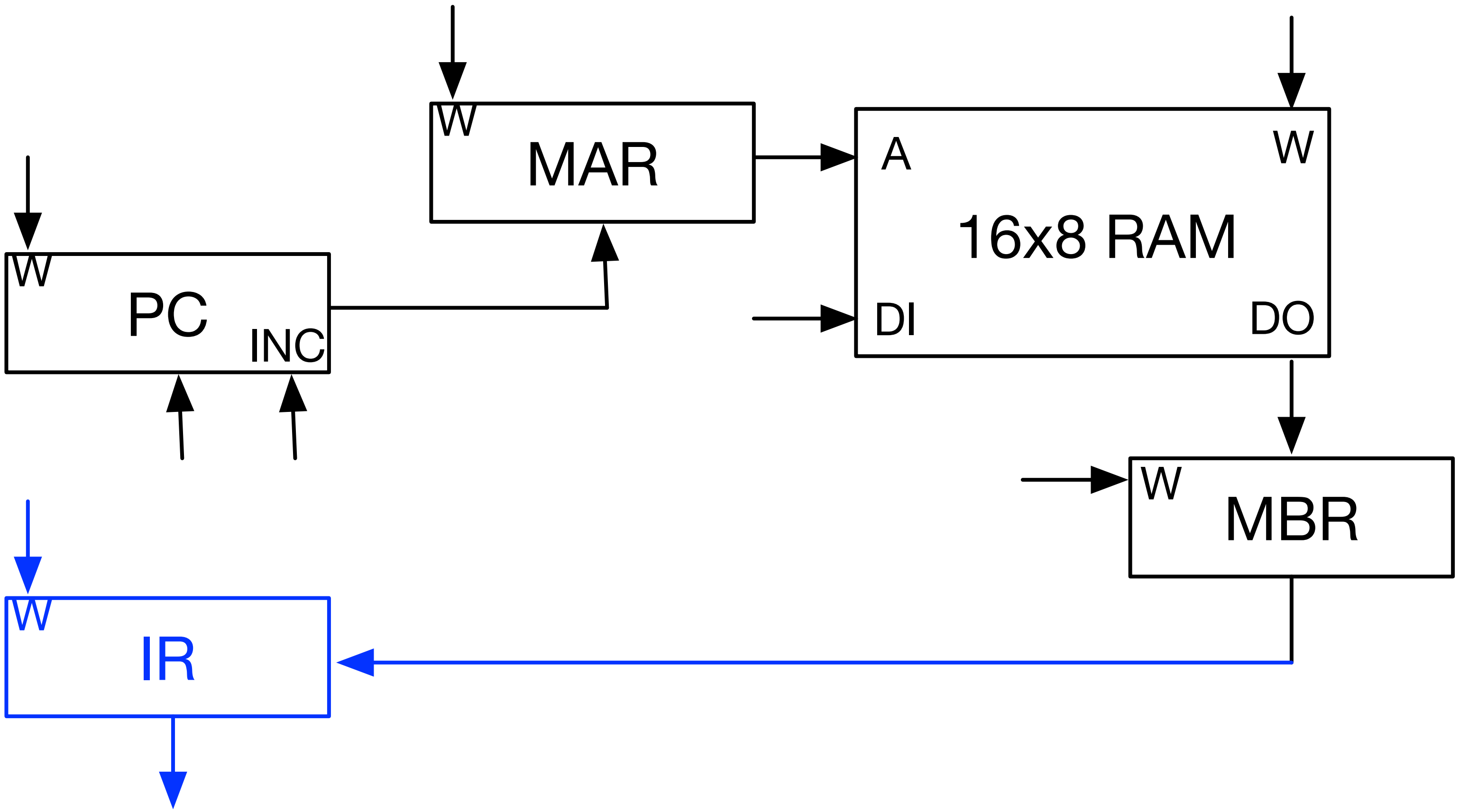
instruction fetch

Program Counter



- Program counter contains address of current instruction
- This address needs to be passed to MAR
- Program counter also needs an easy way to be incremented

Instruction Register

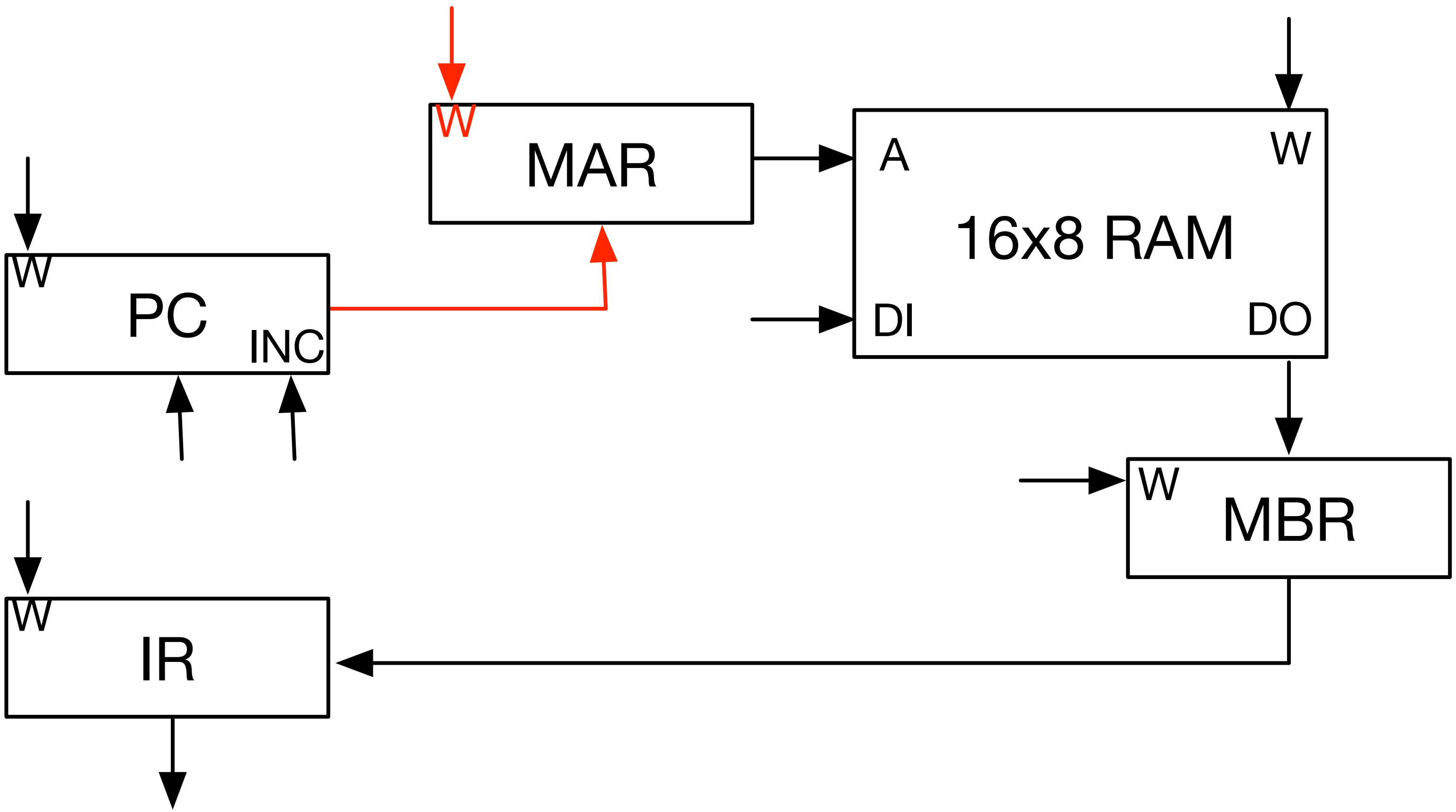


- Content of memory (MBR) is transferred to instruction register

Instruction Fetch

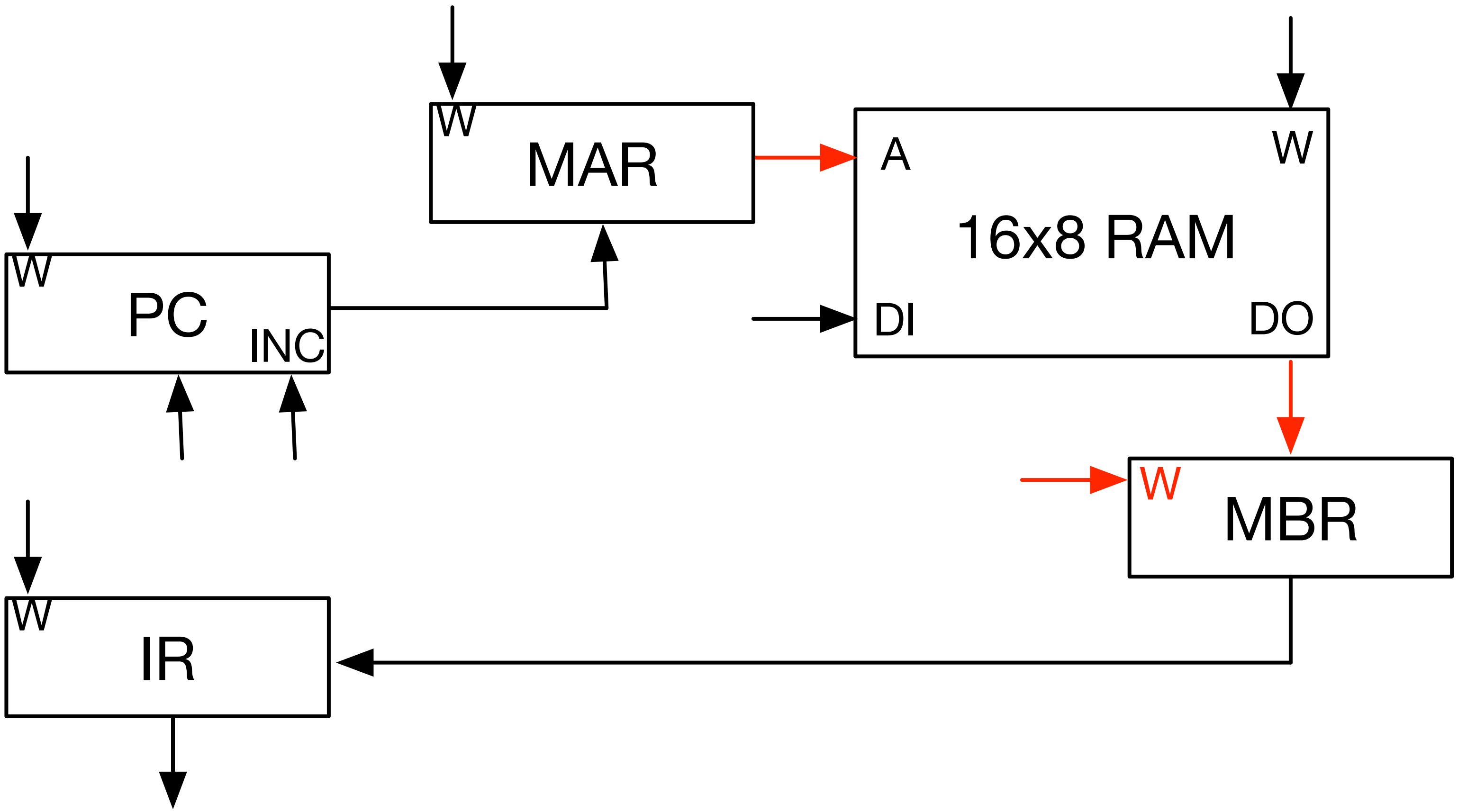
- Program counter (PC) contains address of current instruction
- Step 1: copy PC value to memory access register (MAR)
- Step 2: retrieve address value into memory buffer register (MBR)
- Step 3: copy MBR value into instruction register (IR)
- Step 4: increase PC
- These copy instructions are triggered by write flags

MAR $\leftarrow$ PC



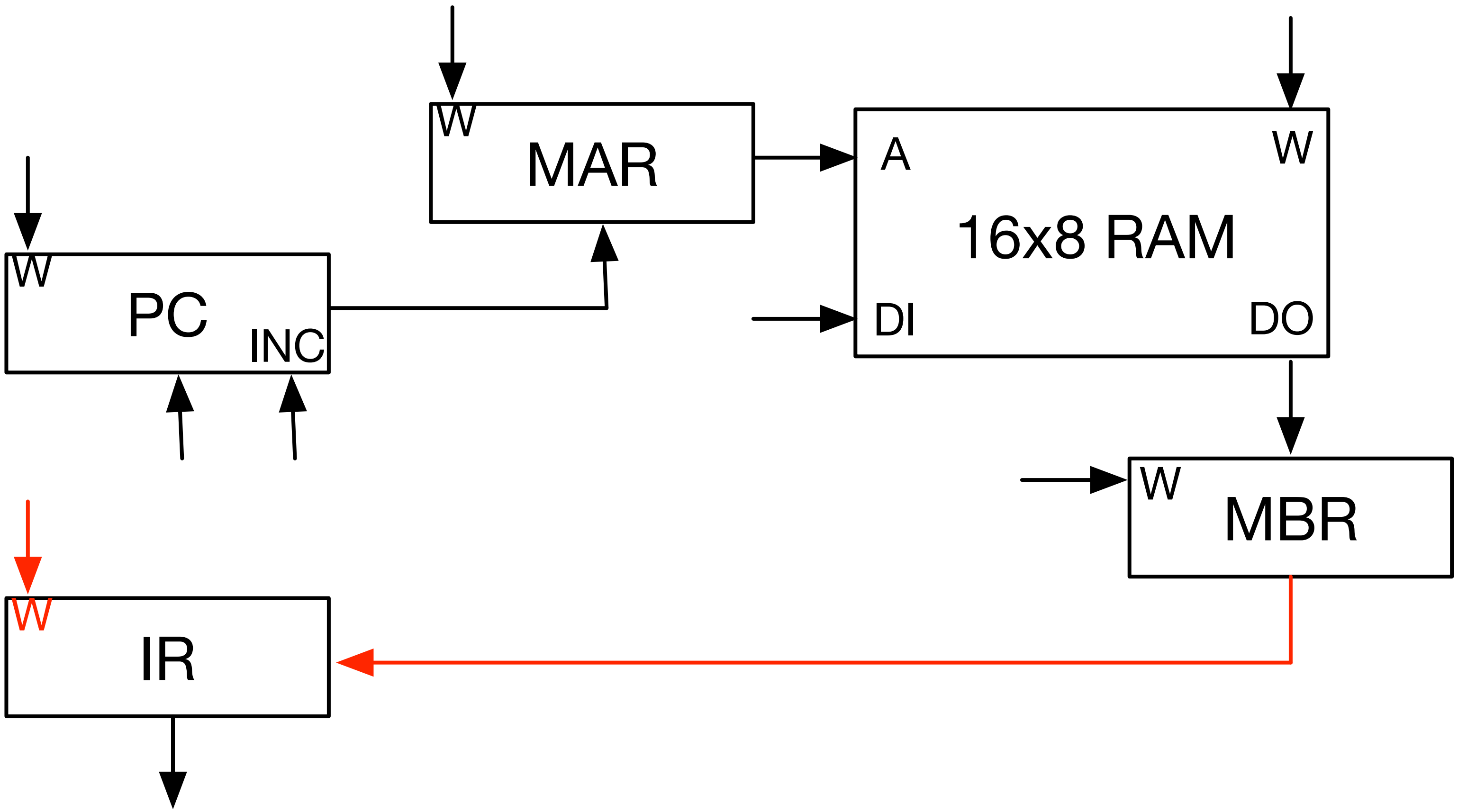
- Copy PC value to memory access register (MAR)

MBR $\leftarrow$ M



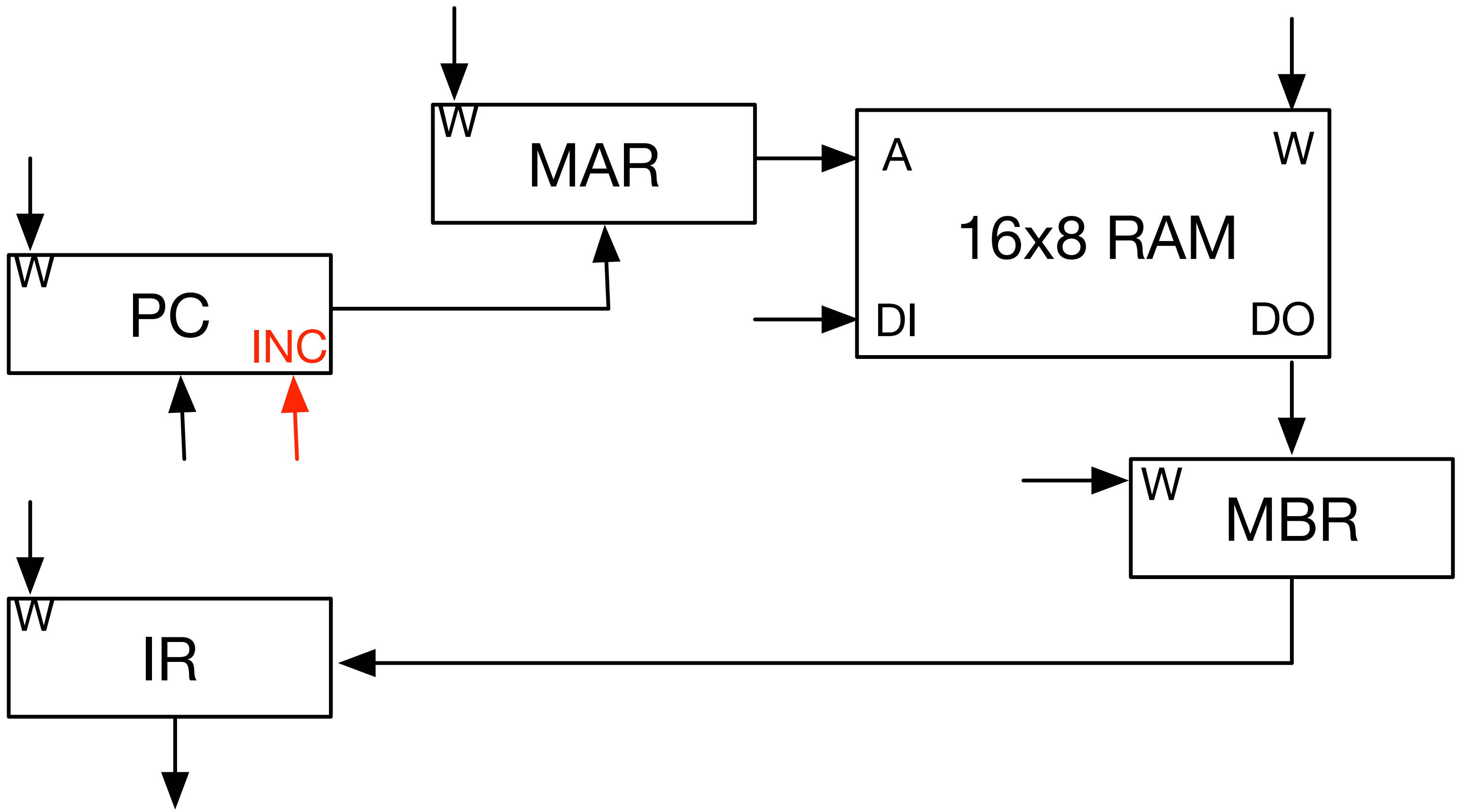
- Retrieve address value into memory buffer register (MBR)

IR ← MBR



- `Copy MBR value into instruction register (IR)`

$\mathbf{PC} \leftarrow \mathbf{PC + 1}$



- Increase PC

Micro Program

- We can write these steps in a *register transfer language*

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

- Execution in time steps t_n triggered by the clock

Parallel Execution

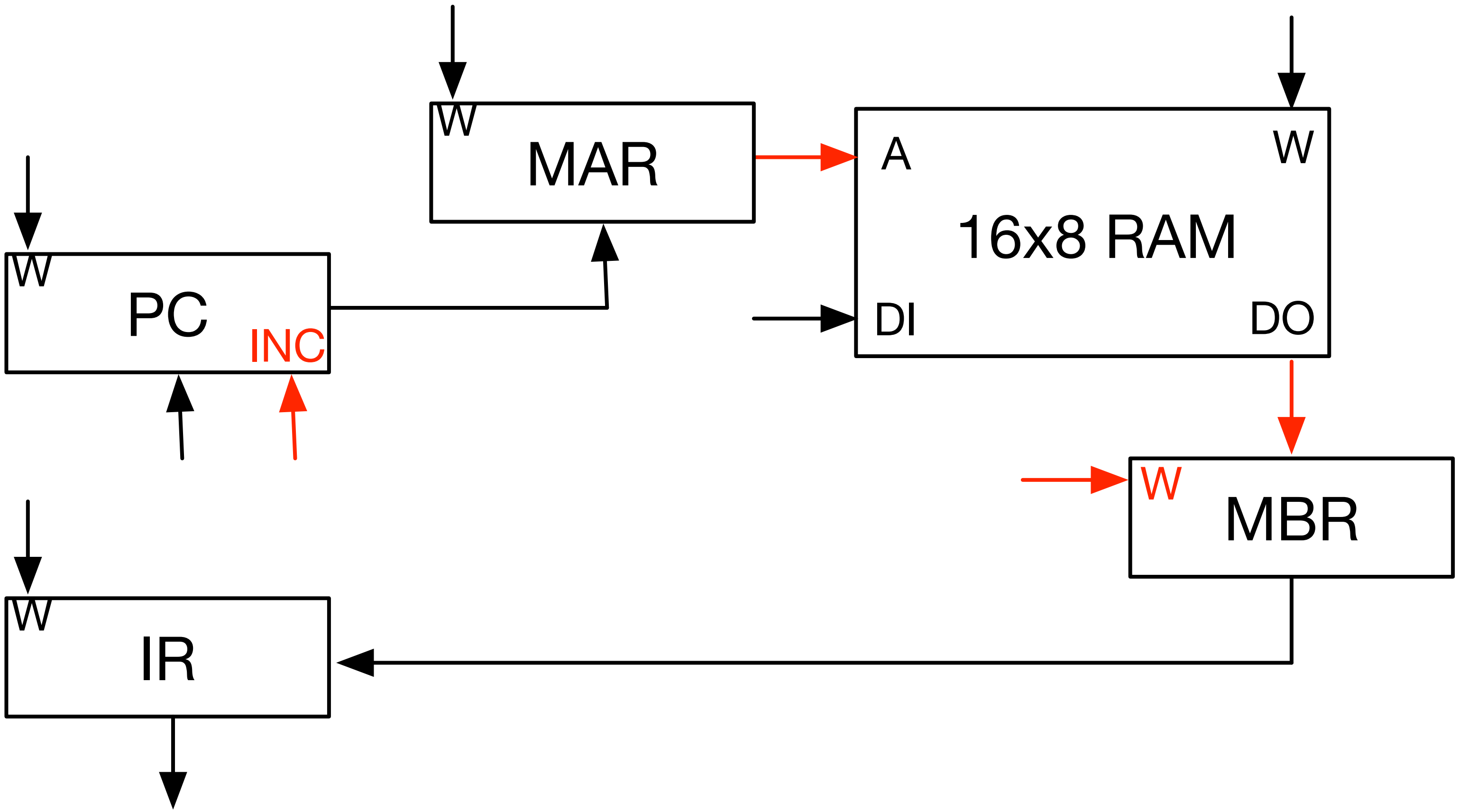
- Increase of the program counter is independent from retrieving data from memory

⇒ These steps can be parallized

- New micro program

Time	Command
t_0	MAR ← PC
t_1	MBR ← M, PC ← PC + 1
t_2	IR ← MBR

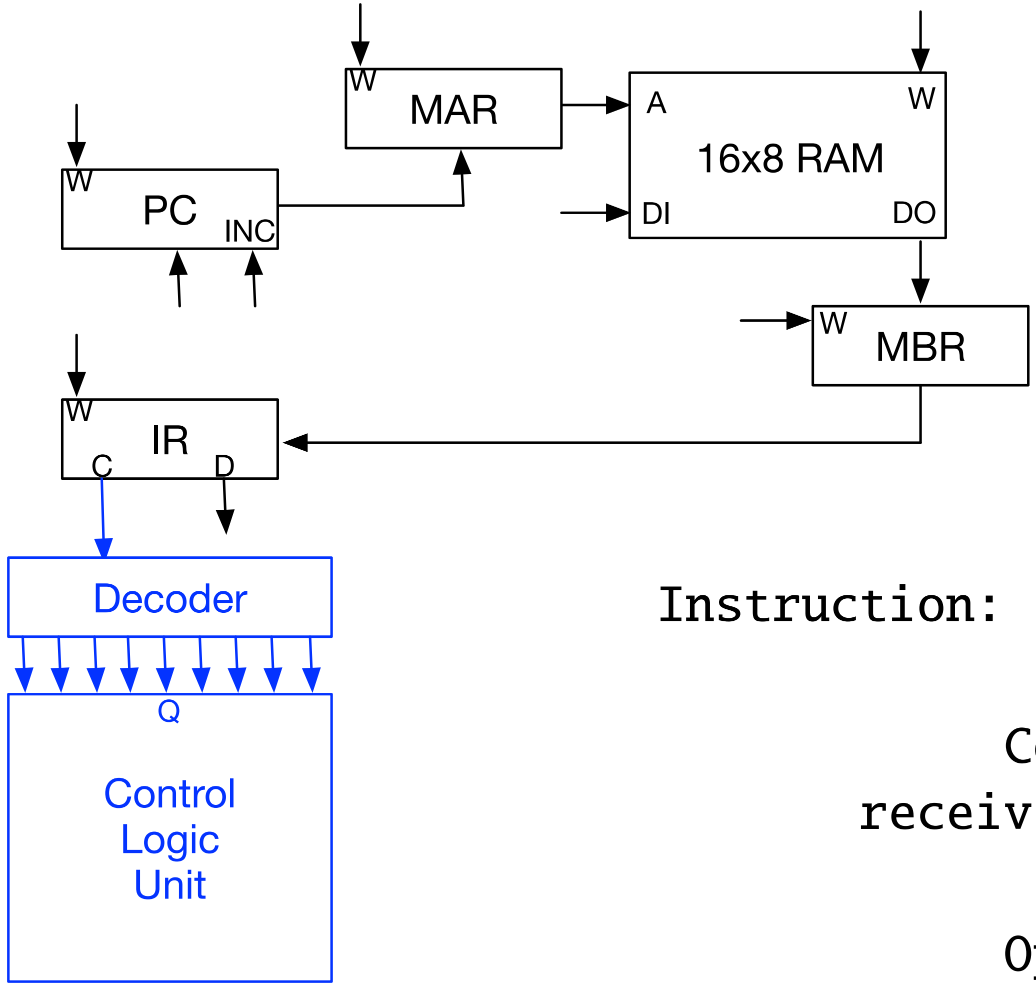
$\mathbf{MBR} \leftarrow \mathbf{M},\ \mathbf{PC} \leftarrow \mathbf{PC + 1}$



- Parallel execution of memory retrieve and program counter increase

control logic unit

Control Logic Unit

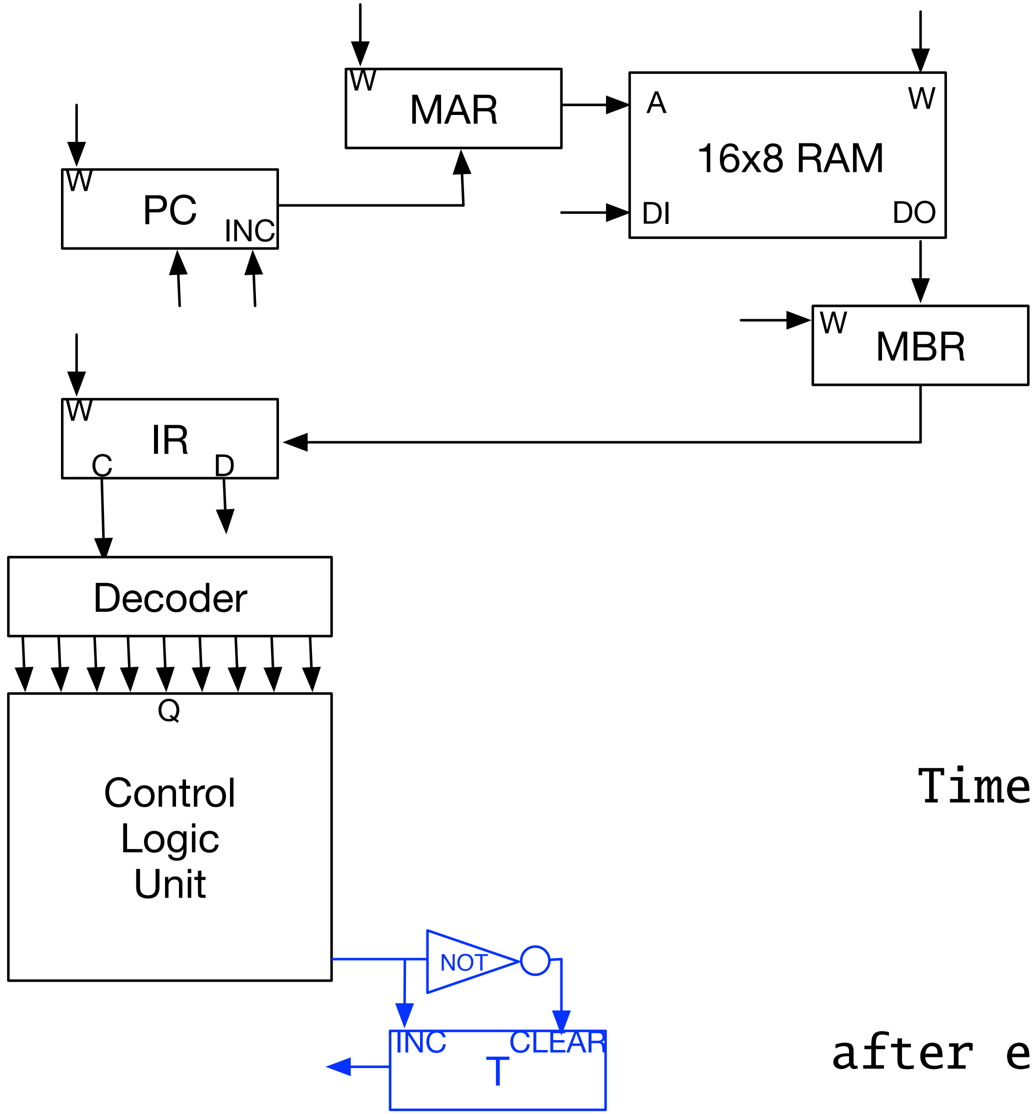


Instruction: op-code and data

Control logic unit
receives operation code

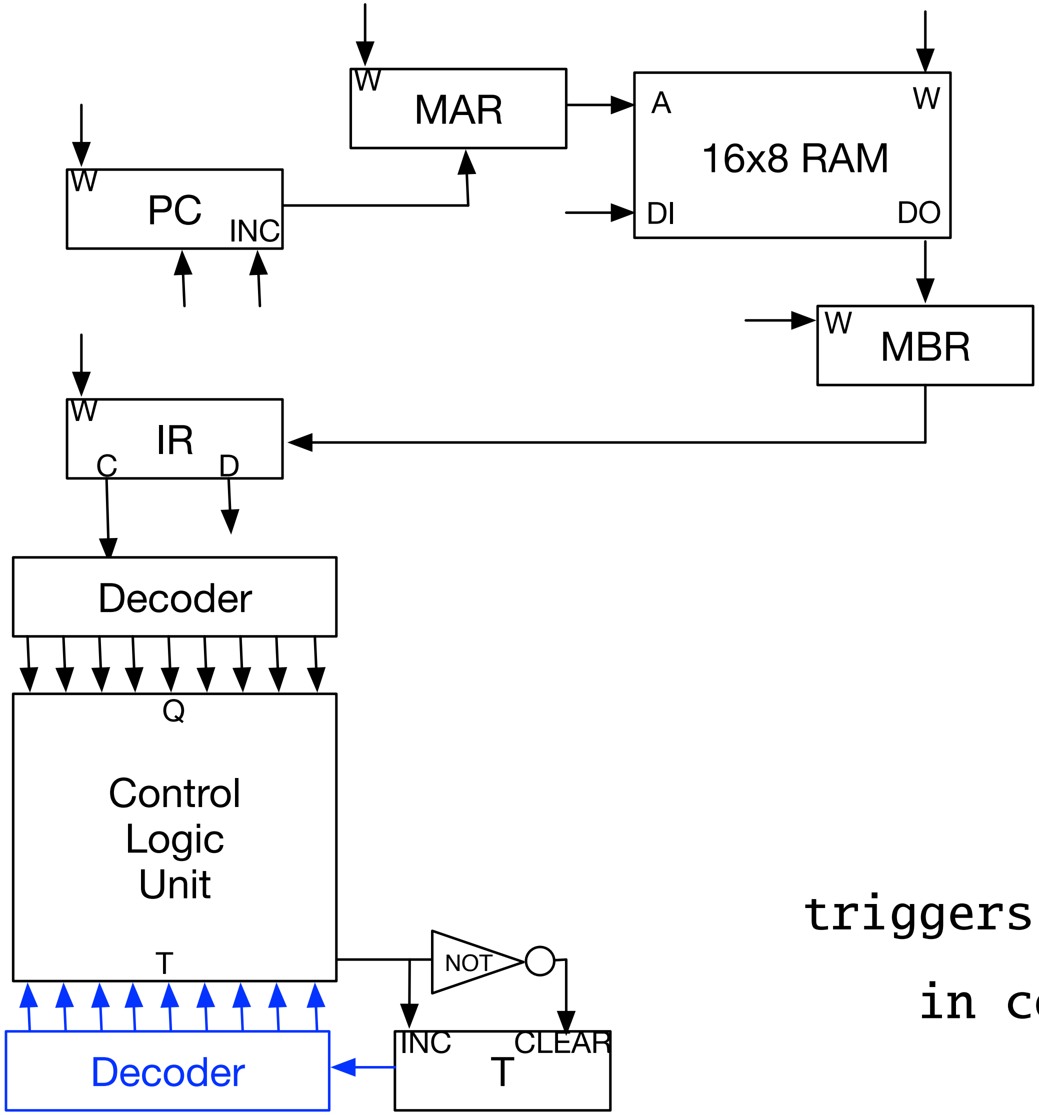
Op-code is decoded
into 9 different operations

Control Logic Unit



Time step needs to be increased or cleared to 0 after each micro command

Control Logic Unit



Time step

triggers one command line

in control logic unit