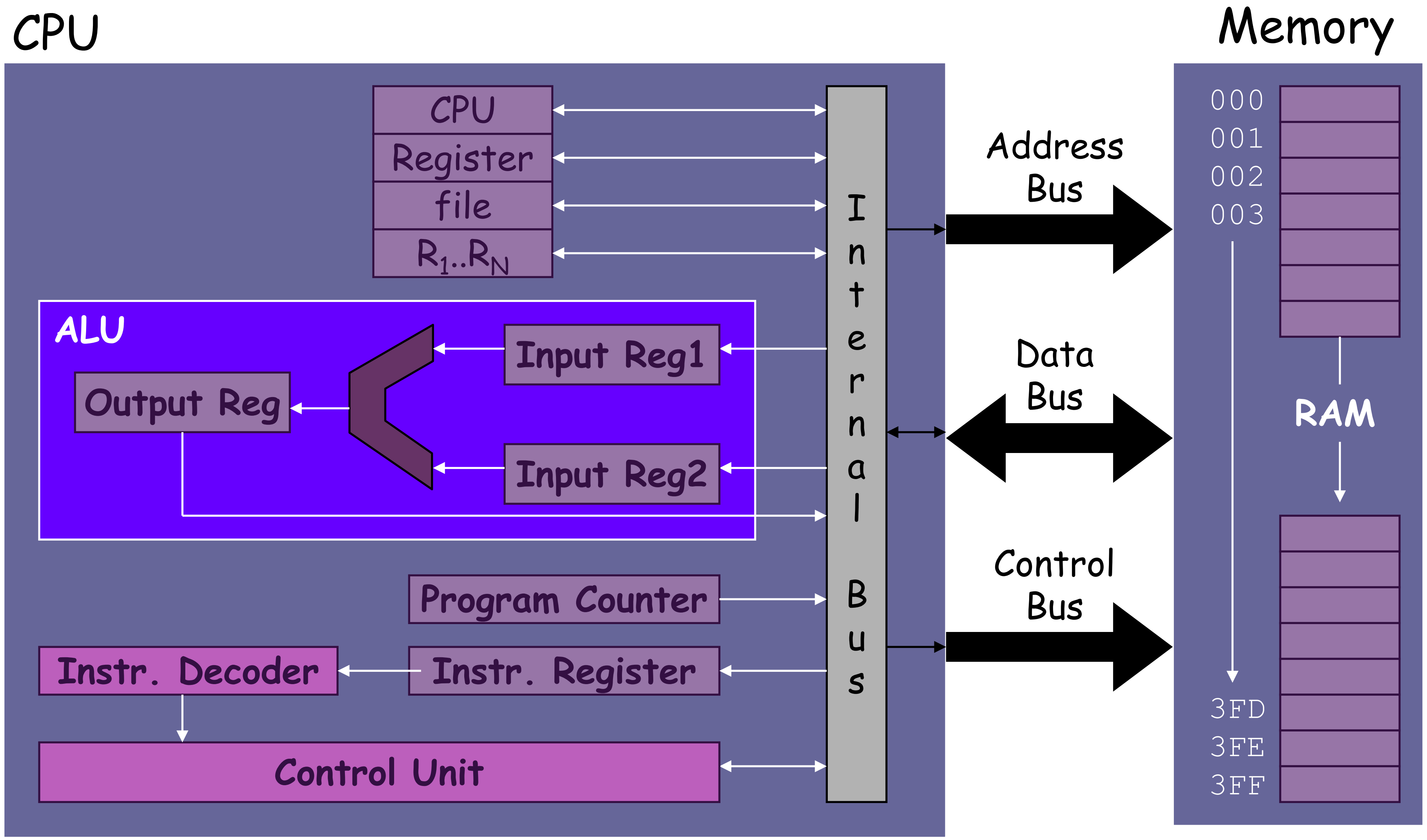
CPU Organisation: *Registers, ALU, Control*



CPU: Central Processing Unit, ALU: Arithmetic and Logic Unit

Fetch-Execute Cycle

- Fetch the ***Instruction*** ***(address in Program Counter PC)***
- Increment ***PC*** ***(prepared to get next instruction)***
- Decode the ***Instruction*** ***(find out tasks to do)***
- Fetch the ***Operands*** ***(data needed for the tasks)***
- Execute the ***Operation*** ***(do the tasks, may involve ALU)***
- Store the ***Results*** ***(in a register or in memory)***
- **Repeat** Forever

High/Low-Level Languages, Machine Code

- High-Level Language (e.g. Java, C++, Haskell)

A = B + C	Assignment Statement

- Low-Level Language: Assembly Language (e.g. Intel IA-32, PowerPC, ARM etc, Java Bytecode)

```
LOAD    R2,  B          R2   = M[b]
ADD     R2,  C          R2   = R2 + M[c]
STORE   R2,  A          M[a] = R2
```

- (Binary) Machine Code

```
0001101000000001        Machine Code
0011101000000010        Instructions
0010101000000000
```

The Toy1 Architecture

- Maximum of **1024 x 16-bit memory words**
  Memory is **Word Addressed**

- **Two's Complement** Integer Representation

- **4 General Purpose Registers** (16-bit) : **R0, R1, R2, R3**

- Up to **16 "Instructions"**, e.g. **LOAD, ADD, STORE**

Toy1 Instruction Set

- **LOAD Register , [MemoryAddress]**
  Register = Memory [MemoryAddress]

- **STORE Register , [MemoryAddress]**
  Memory [MemoryAddress] = Register

- **ADD Register , [MemoryAddress]**
  Register = Register + Memory [MemoryAddress]

- **SUB Register , [MemoryAddress]**
  Register = Register - Memory [MemoryAddress]

Toy1 Instruction Format

Assembly Instruction e.g. ADD R2, C

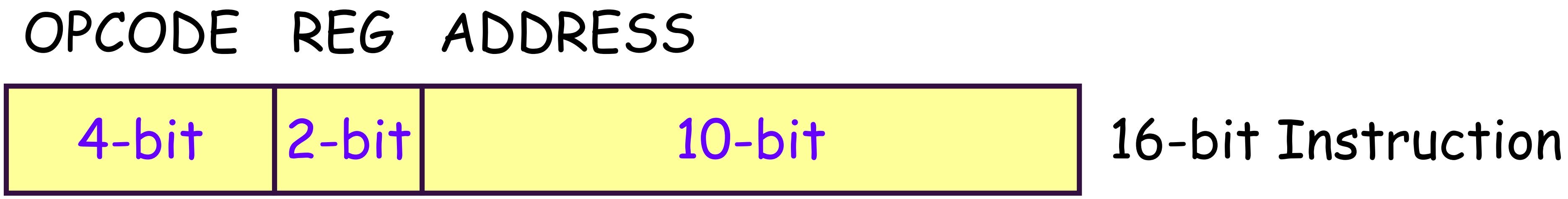
Machine Code

OPCODE	REG	ADDRESS
4-bit	2-bit	10-bit

Instruction Fields

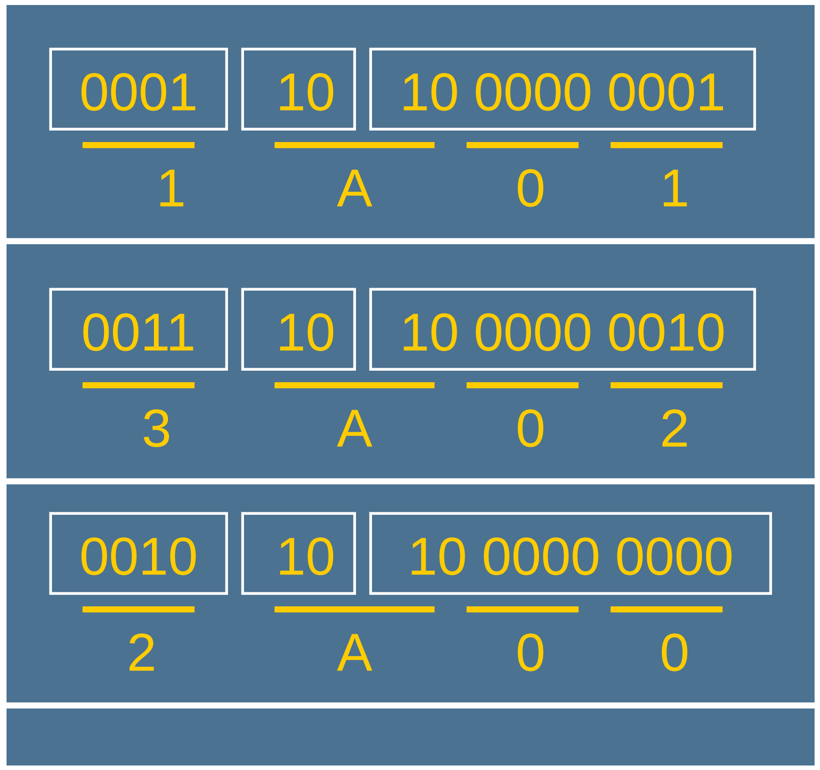
- **OP**eration **CODE** (Selects CPU Instruction)
- **REG**ister (Specifies 1st Operand for Instruction)
- **ADDRESS** (Specifies 2nd Operand for Instruction)

Instruction Field Encoding



➢ **OPCODE**	LOAD	0001
(4-bit)	STORE	0010
	ADD	0011
	SUB	0100
➢ **REG**	Register 0	00
(2-bit)	Register 1	01
	Register 2	10
	Register 3	11
➢ **ADDRESS**	10-bit Memory Word Address	

Memory Placement (Program)



Assembly Instruction	Machine Instruction OP	REG	ADDRESS		Memory Address	
LOAD R2, [201H]	0001	10	10 0000 0001	1 A 0 1	00 1000 0000	0 8 0 H
ADD R2, [202H]	0011	10	10 0000 0010	3 A 0 2	00 1000 0001	0 8 1 H
STORE R2, [200H]	0010	10	10 0000 0000	2 A 0 0	00 1000 0010	0 8 2 H

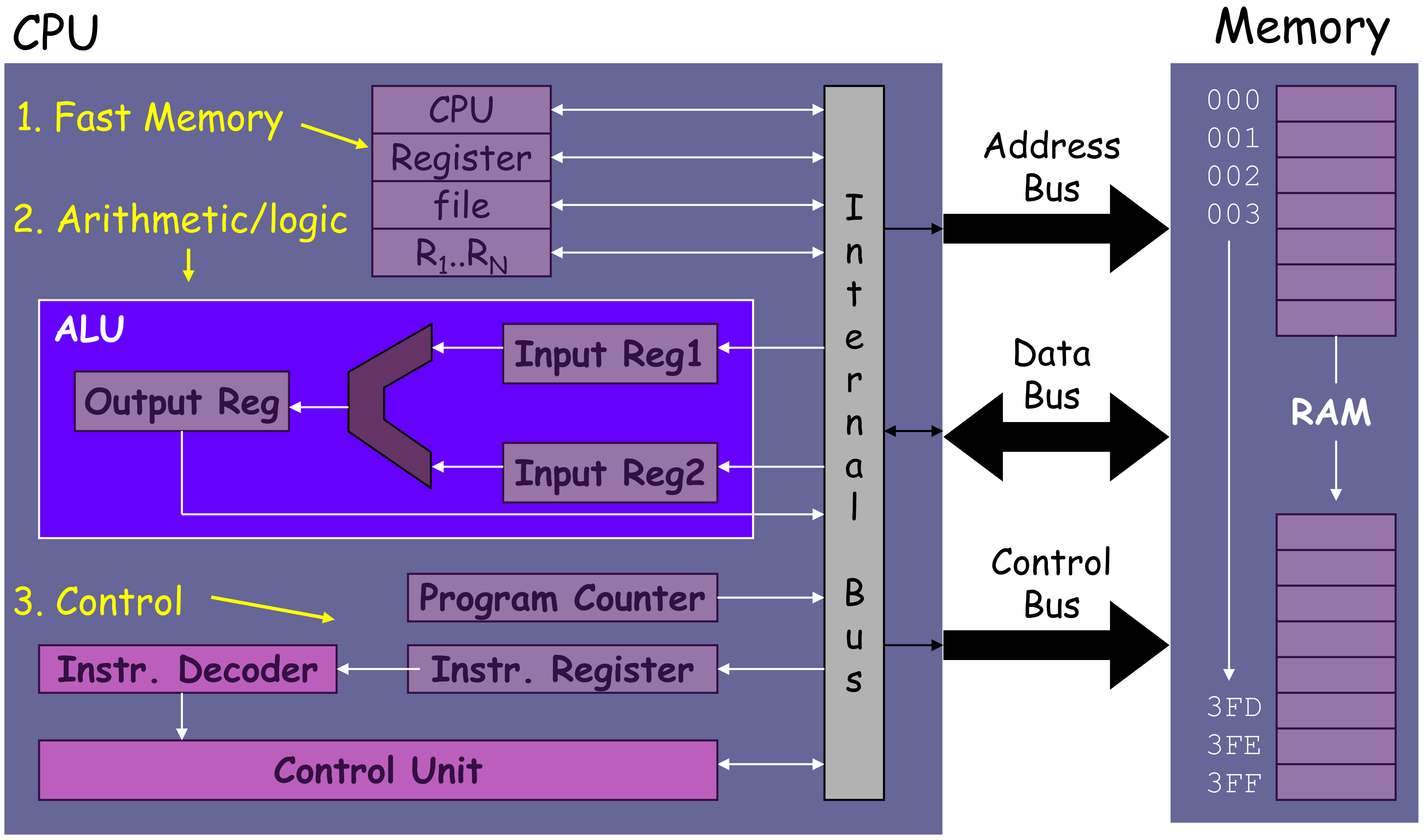
MEMORY

Memory Placement (Data)

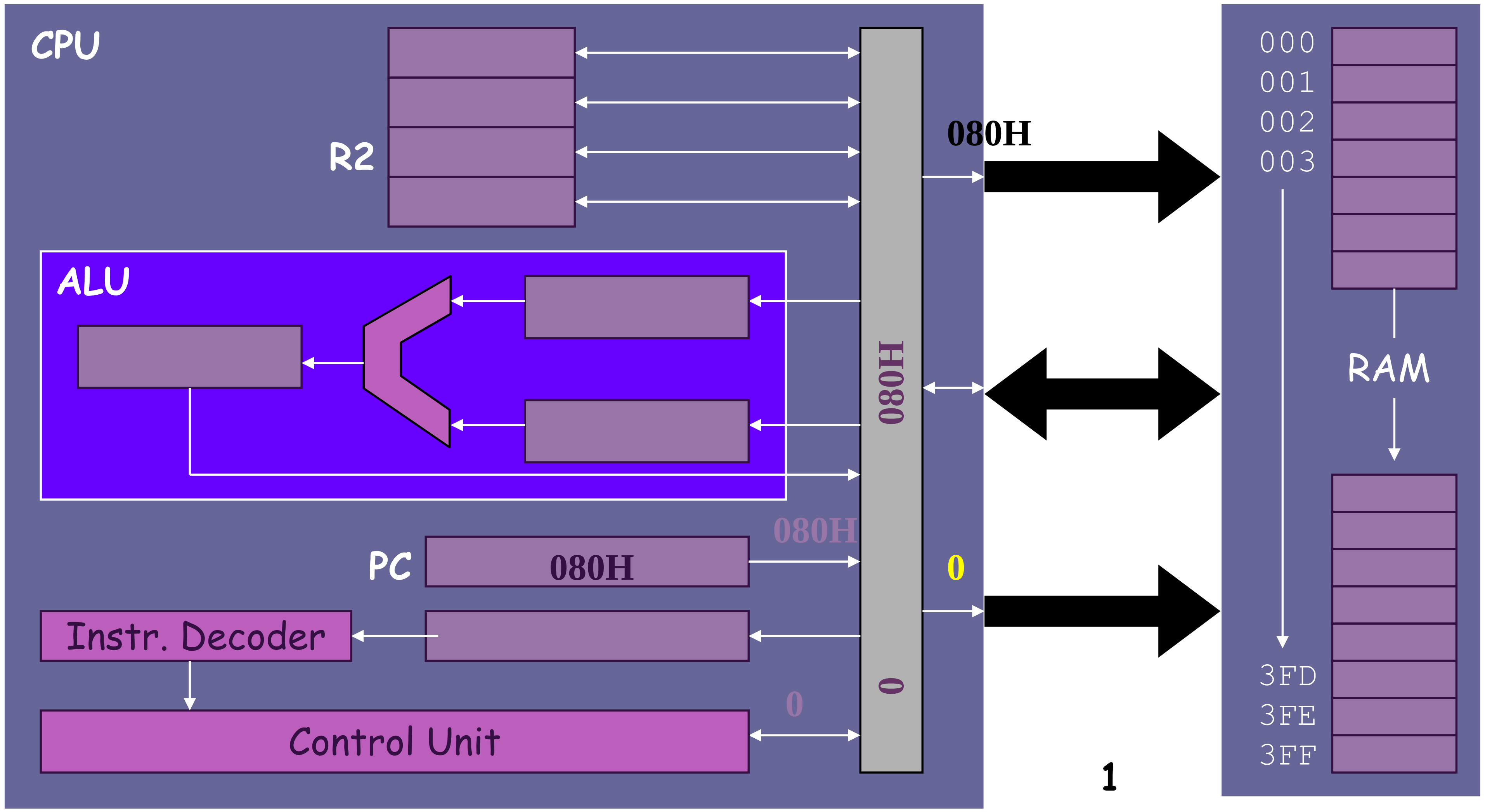
Assembly Instruction	Data	Memory Address
A = 0	0000 0000 0000 0000 0 0 0 0	10 0000 0000 2 0 0 H
B = 9	0000 0000 0000 1001 0 0 0 9	10 0000 0001 2 0 1 H
C = 6	0000 0000 0000 0110 0 0 0 6	10 0000 0010 2 0 2 H

MEMORY

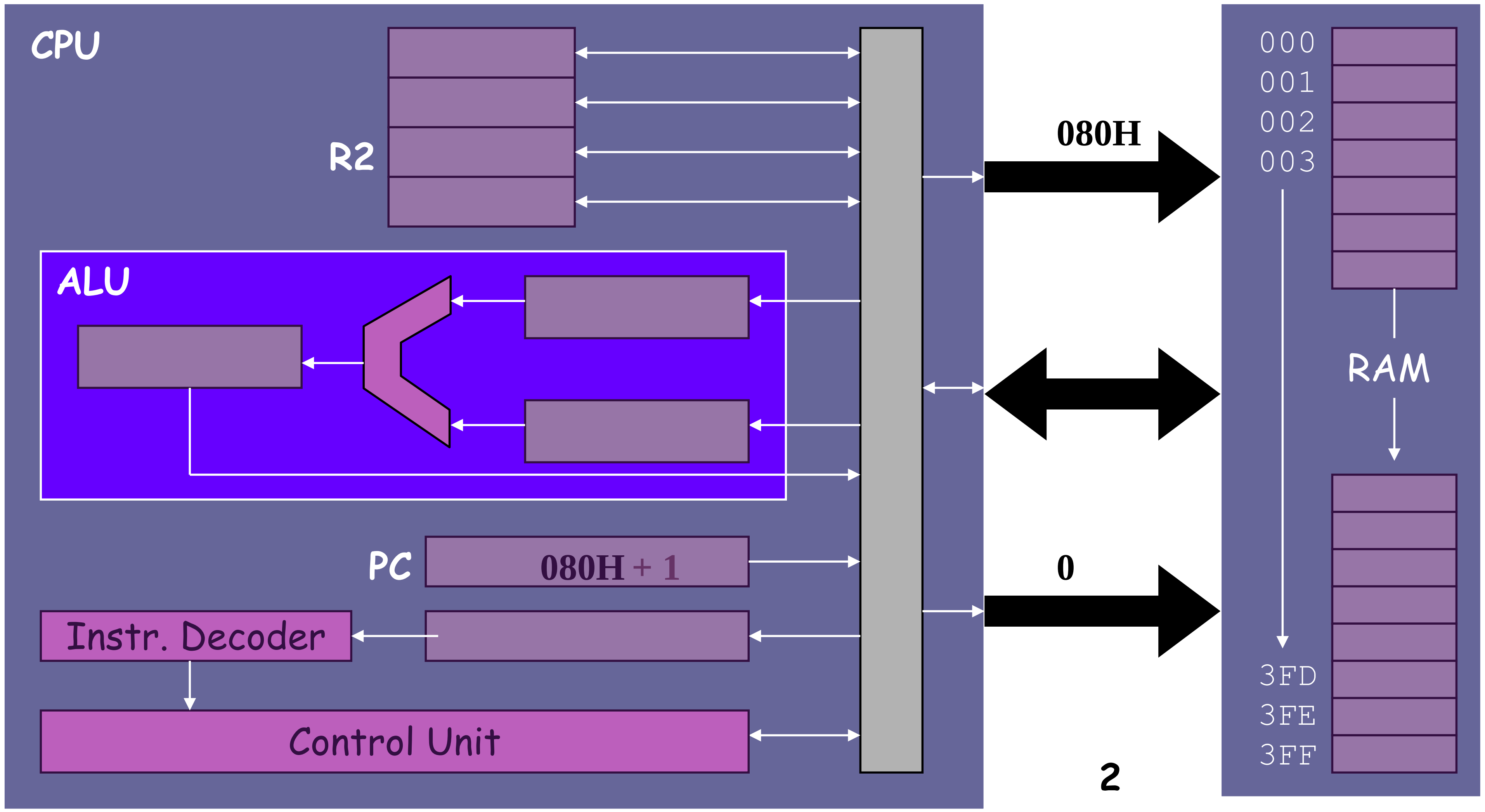
CPU Organisation: Overview



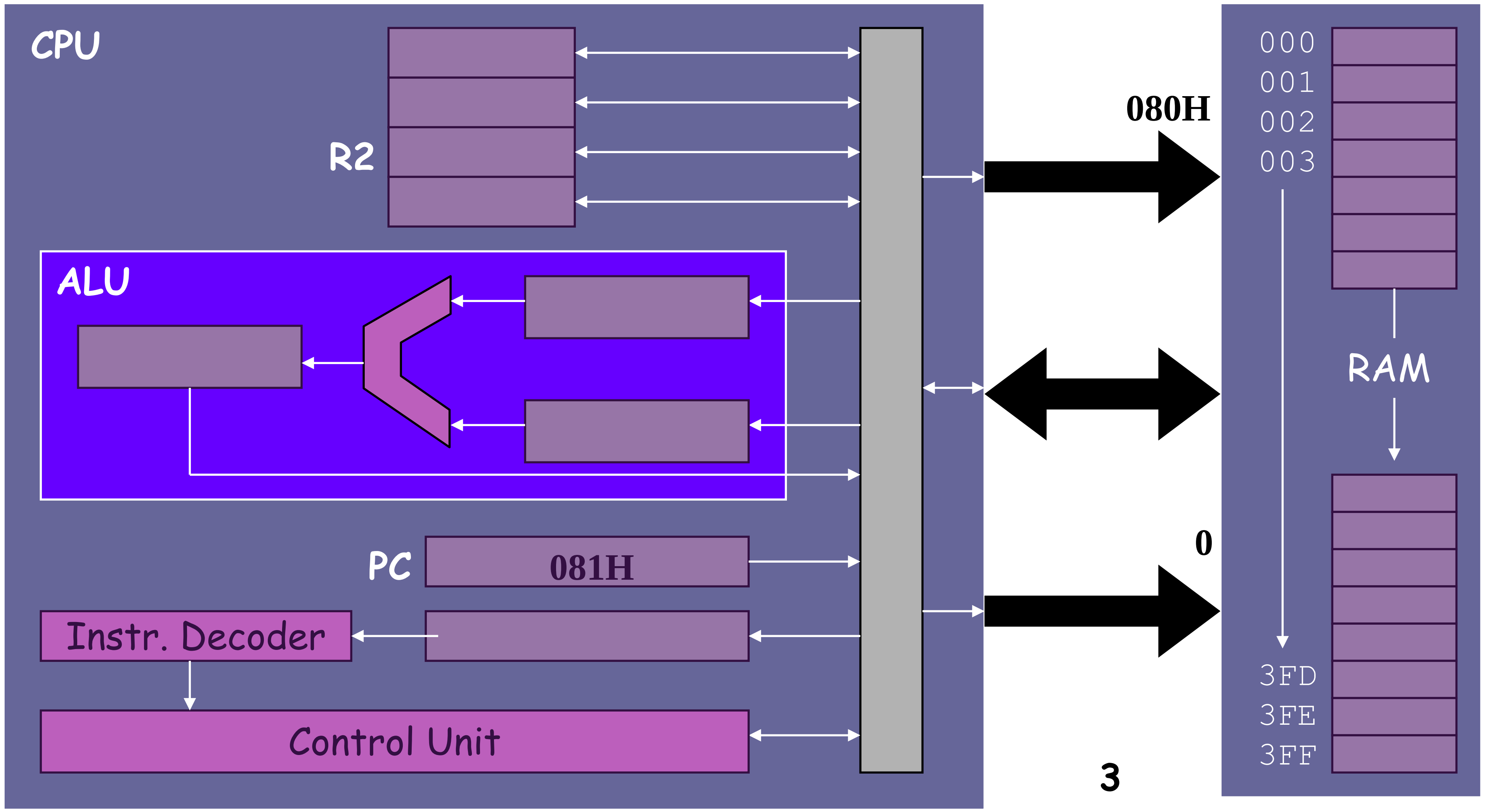
LOAD R2, [201H] R2=Memory[201H]



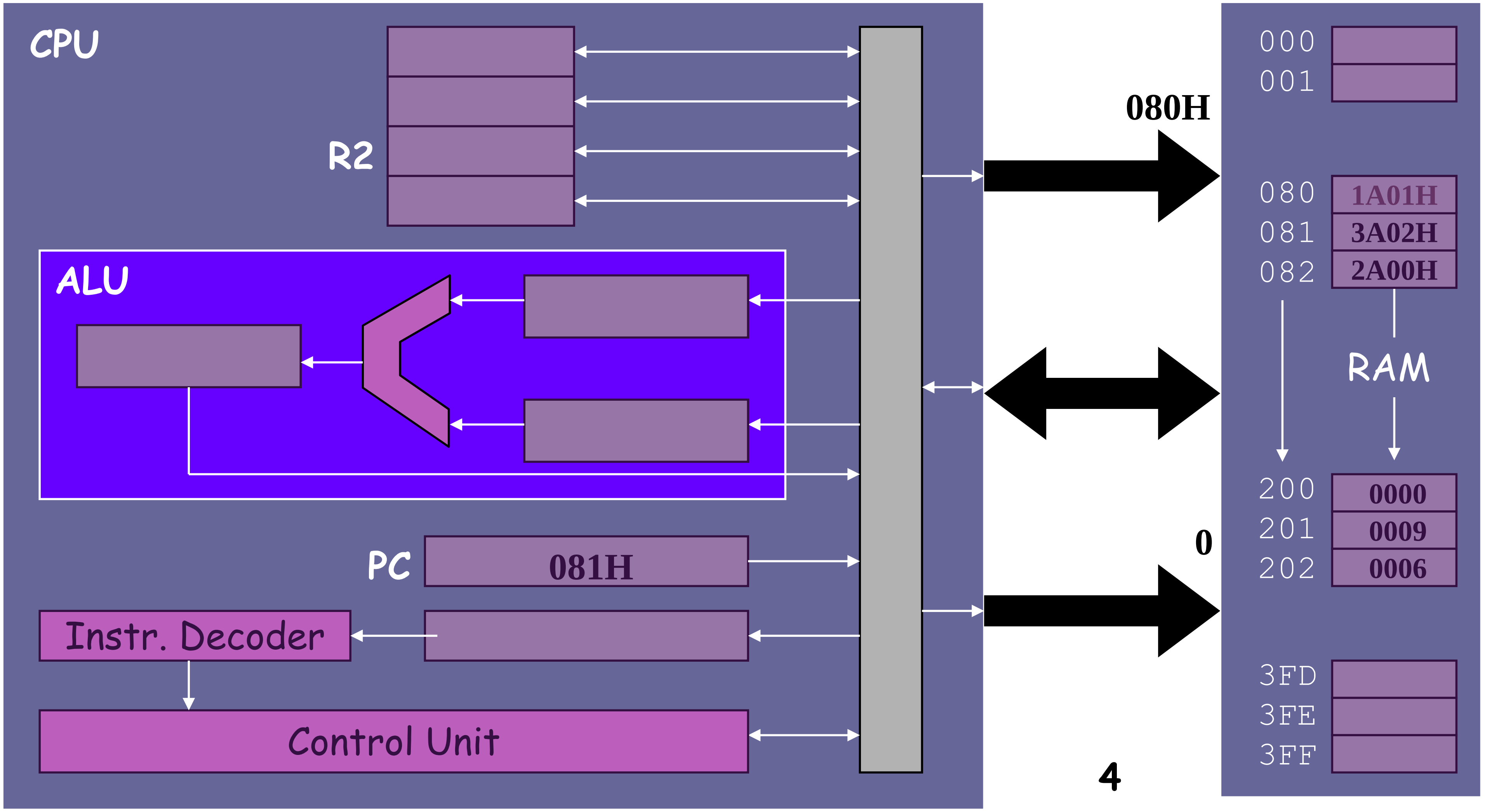
LOAD R2, [201H] R2=Memory[201H]



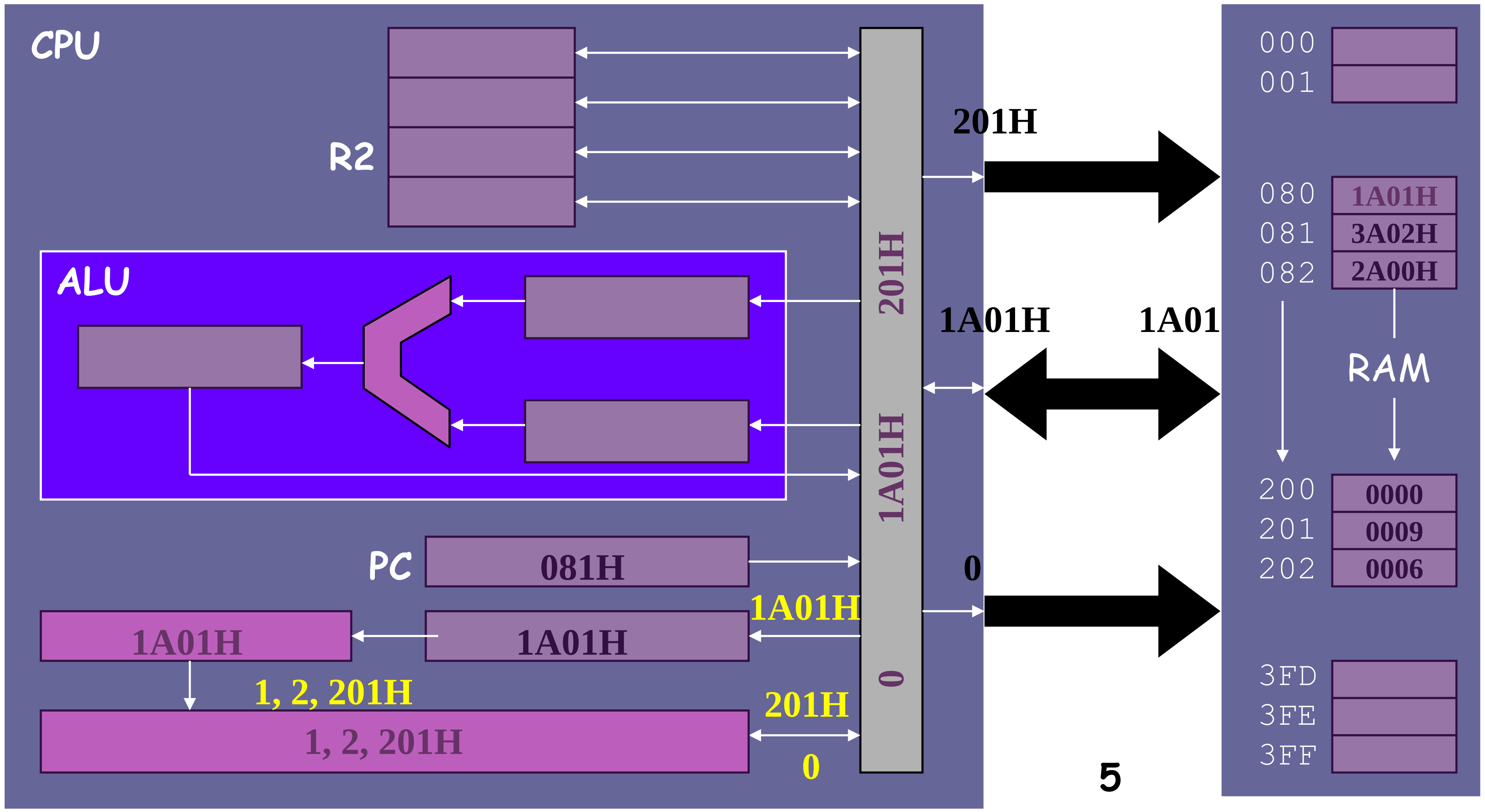
LOAD R2, [201H]   R2=Memory[201H]



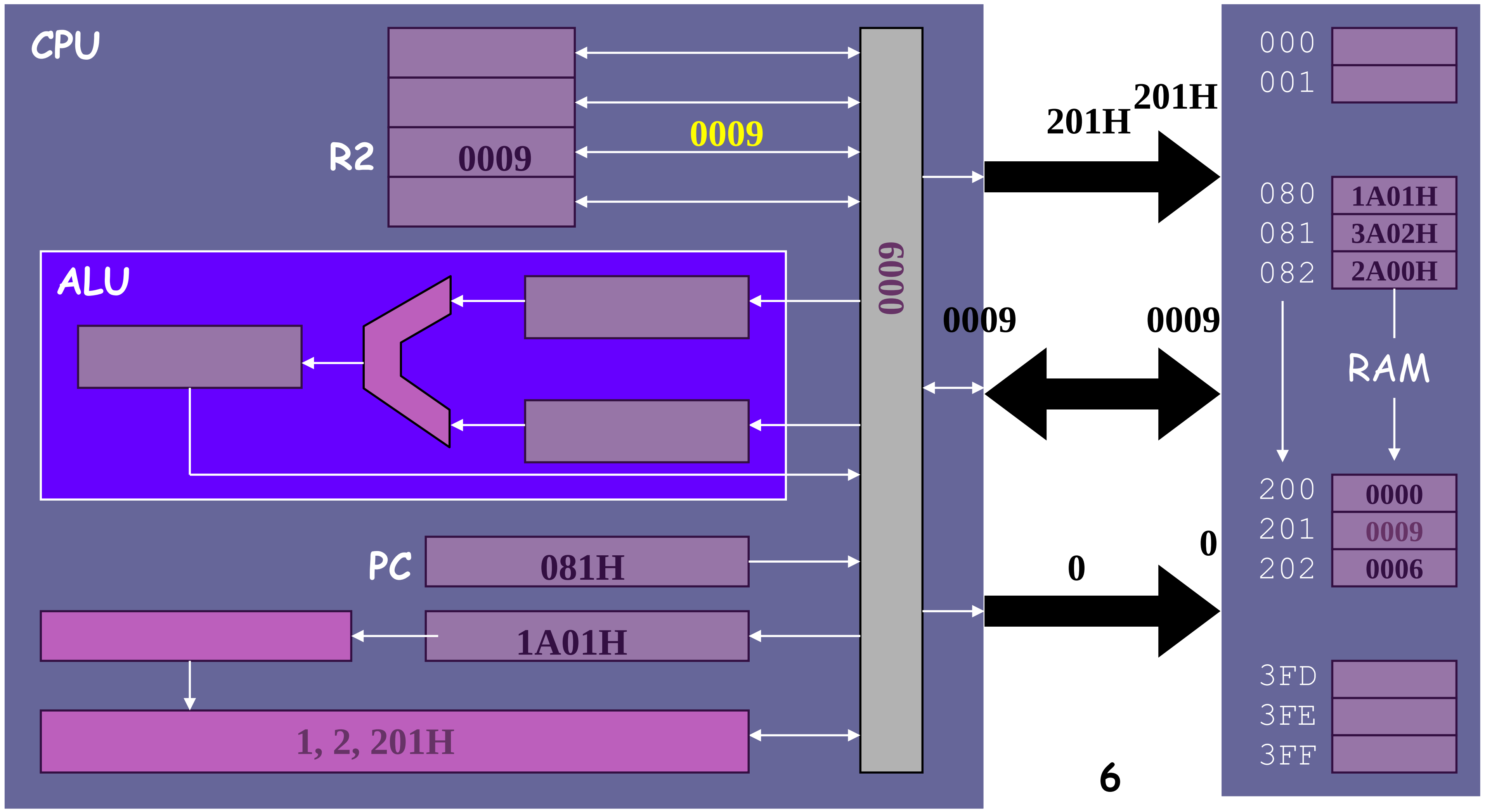
LOAD R2, [201H]   R2=Memory[201H]



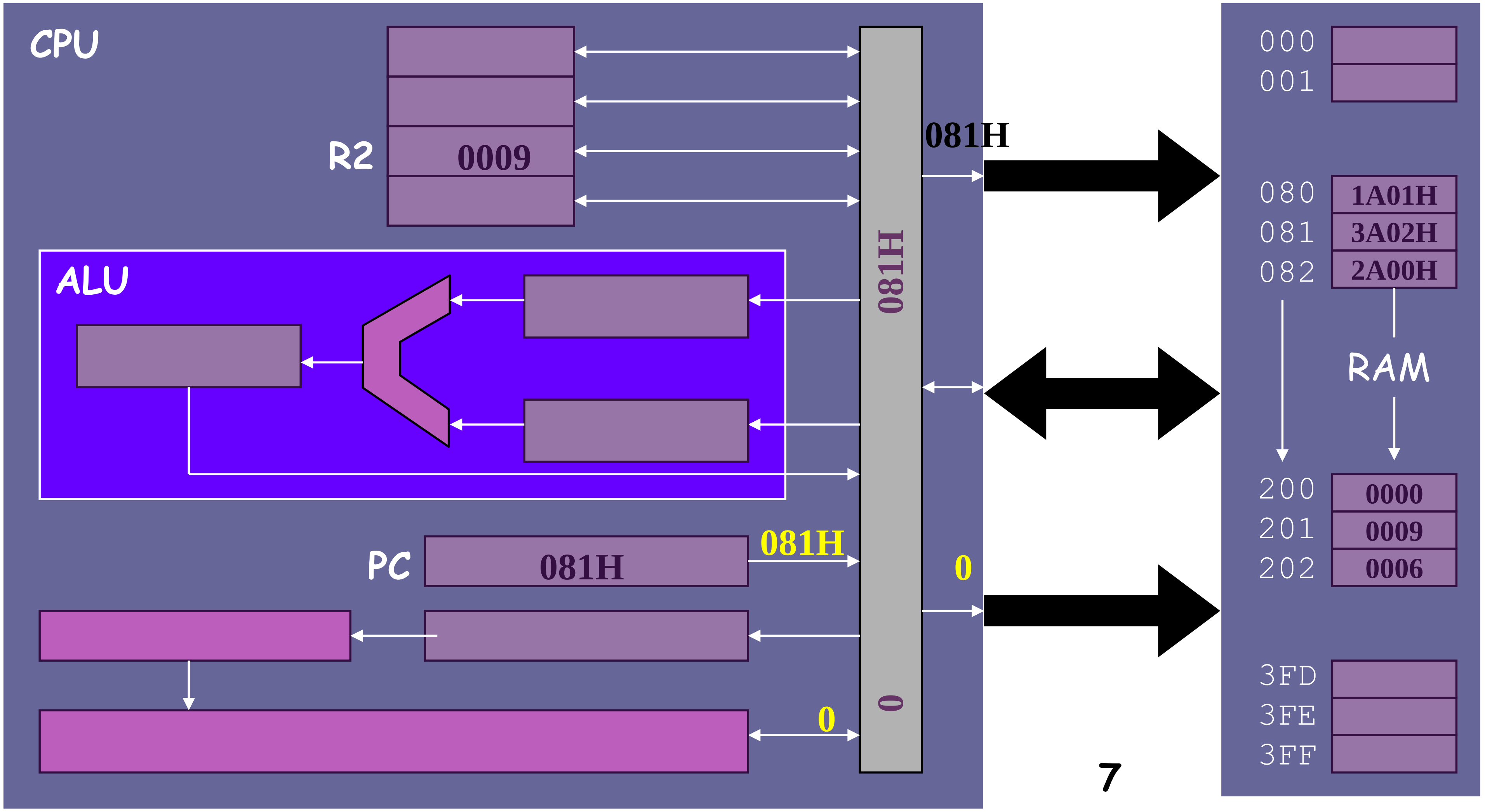
LOAD R2, [201H] R2=Memory[201H]



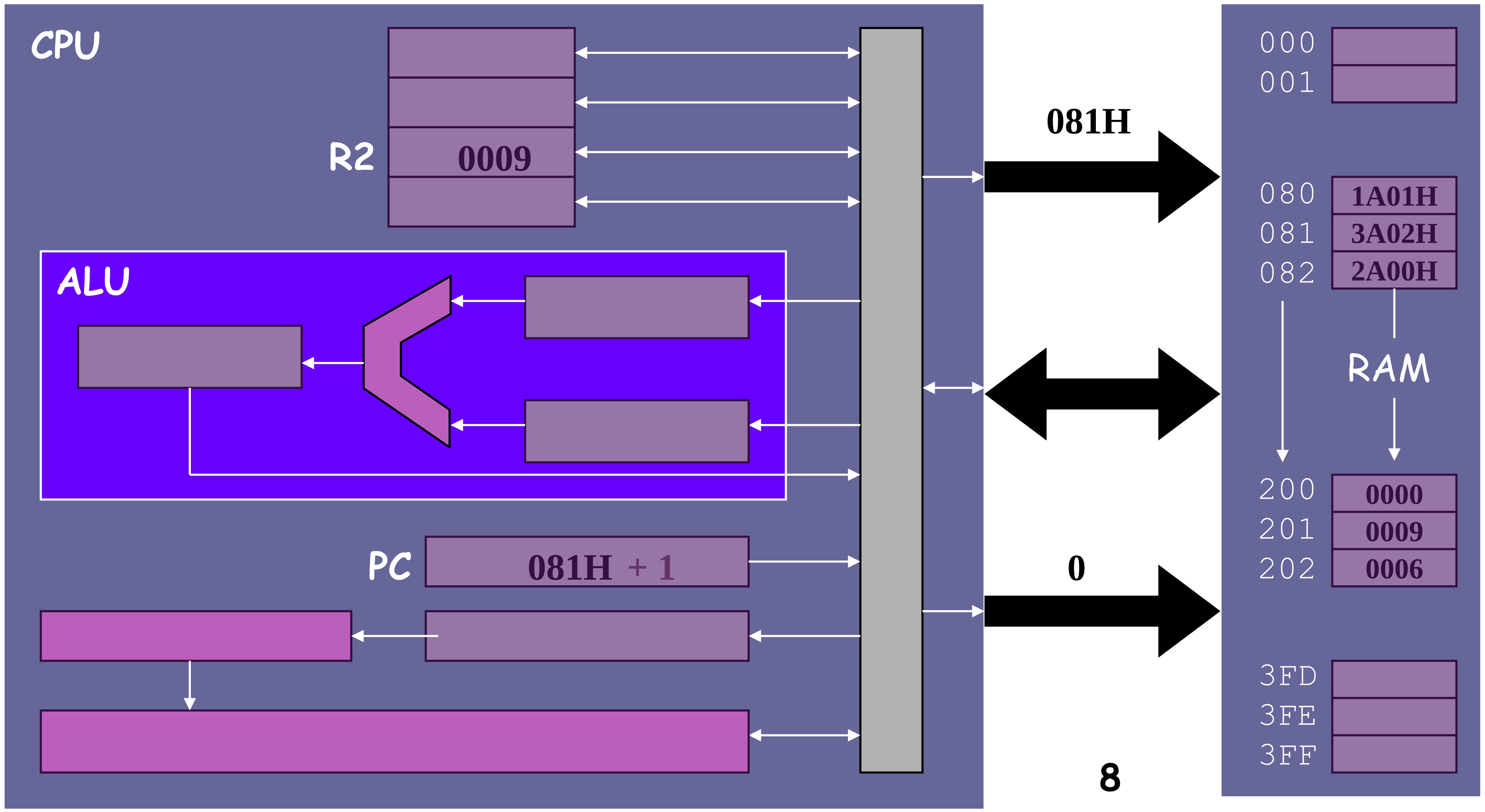
LOAD R2, [201H] R2=Memory[201H]



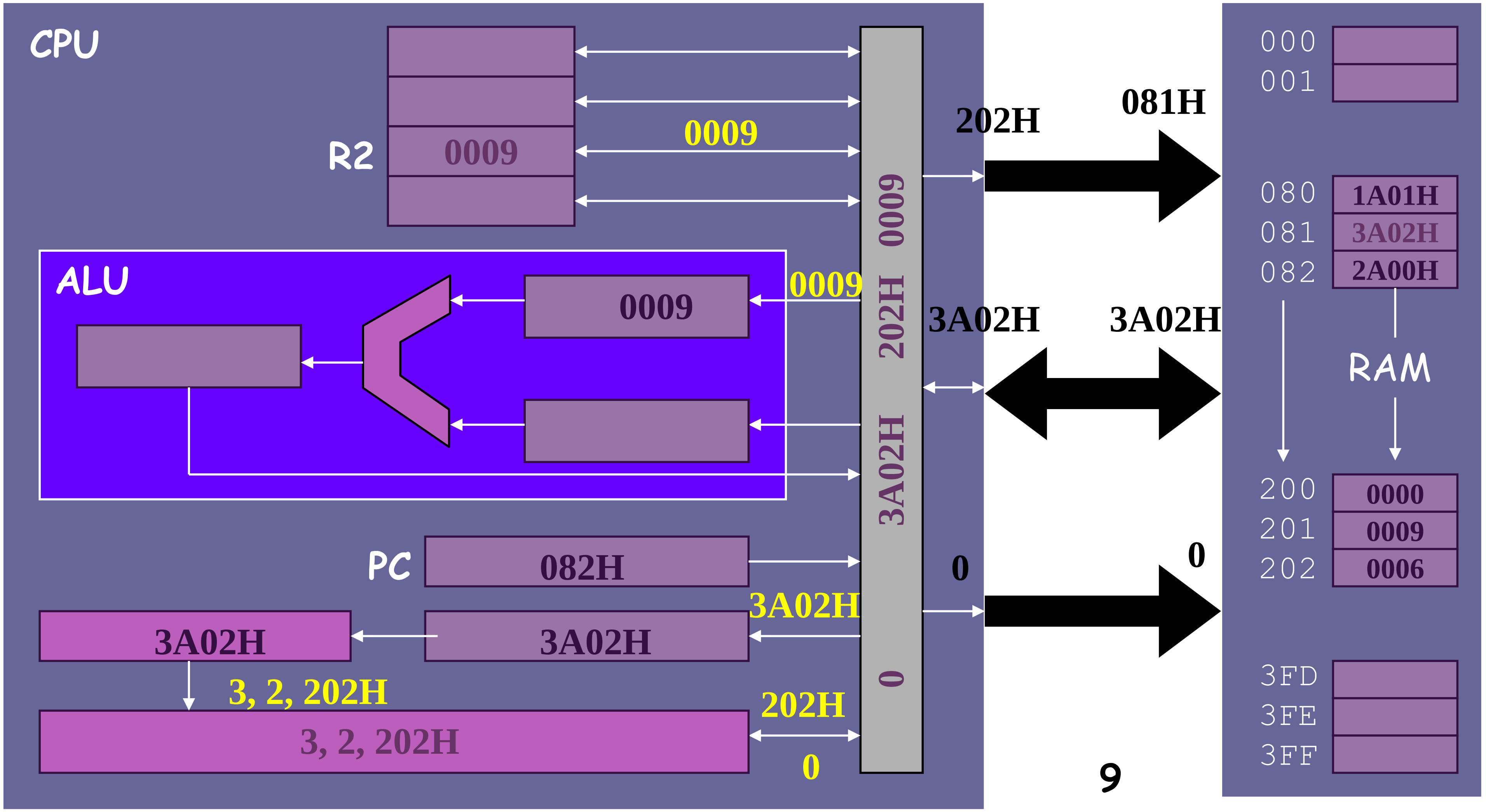
ADD R2, [202H]   R2=R2+Memory[202H]



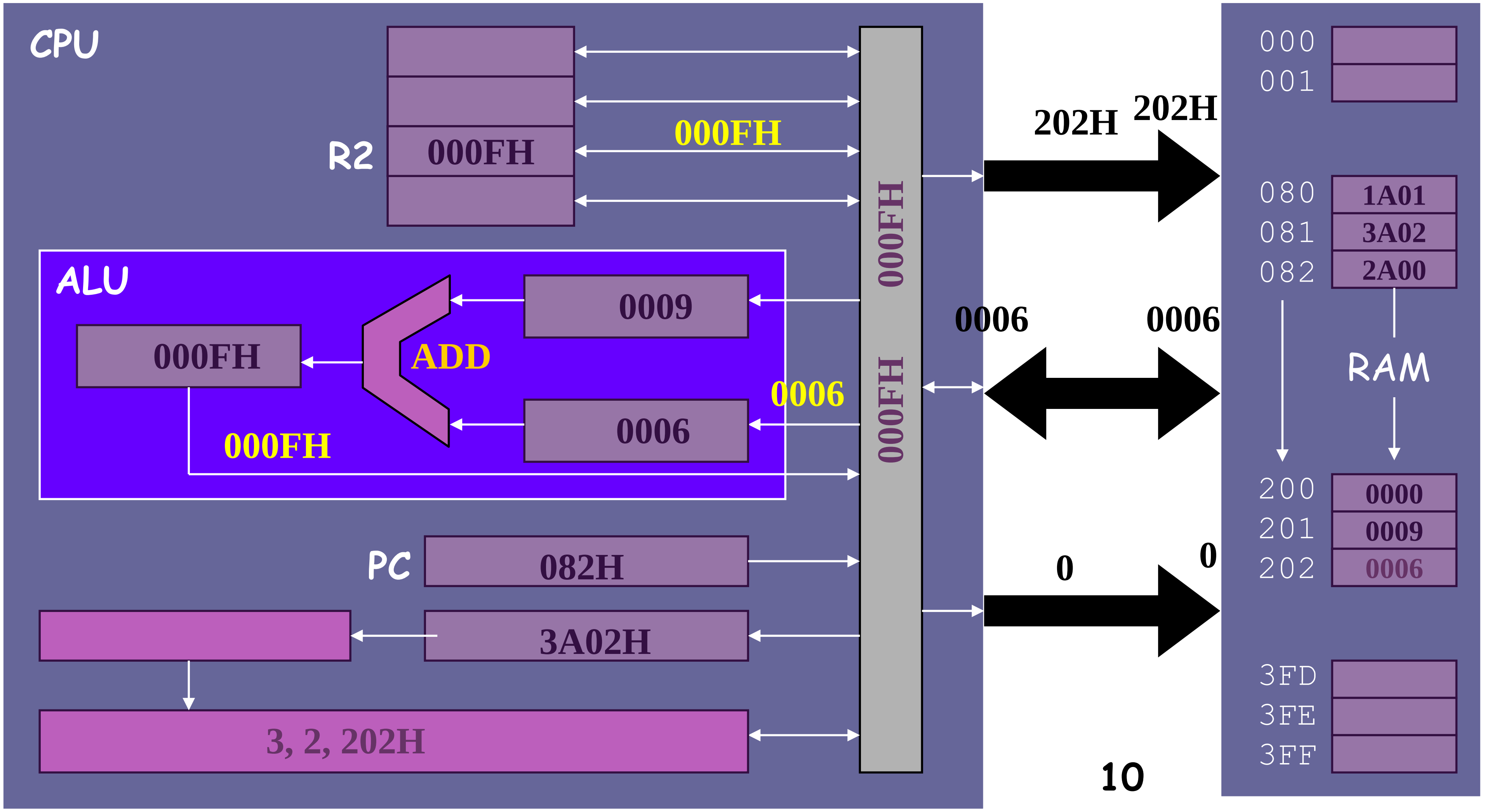
ADD R2, [202H]   R2=R2+Memory[202H]



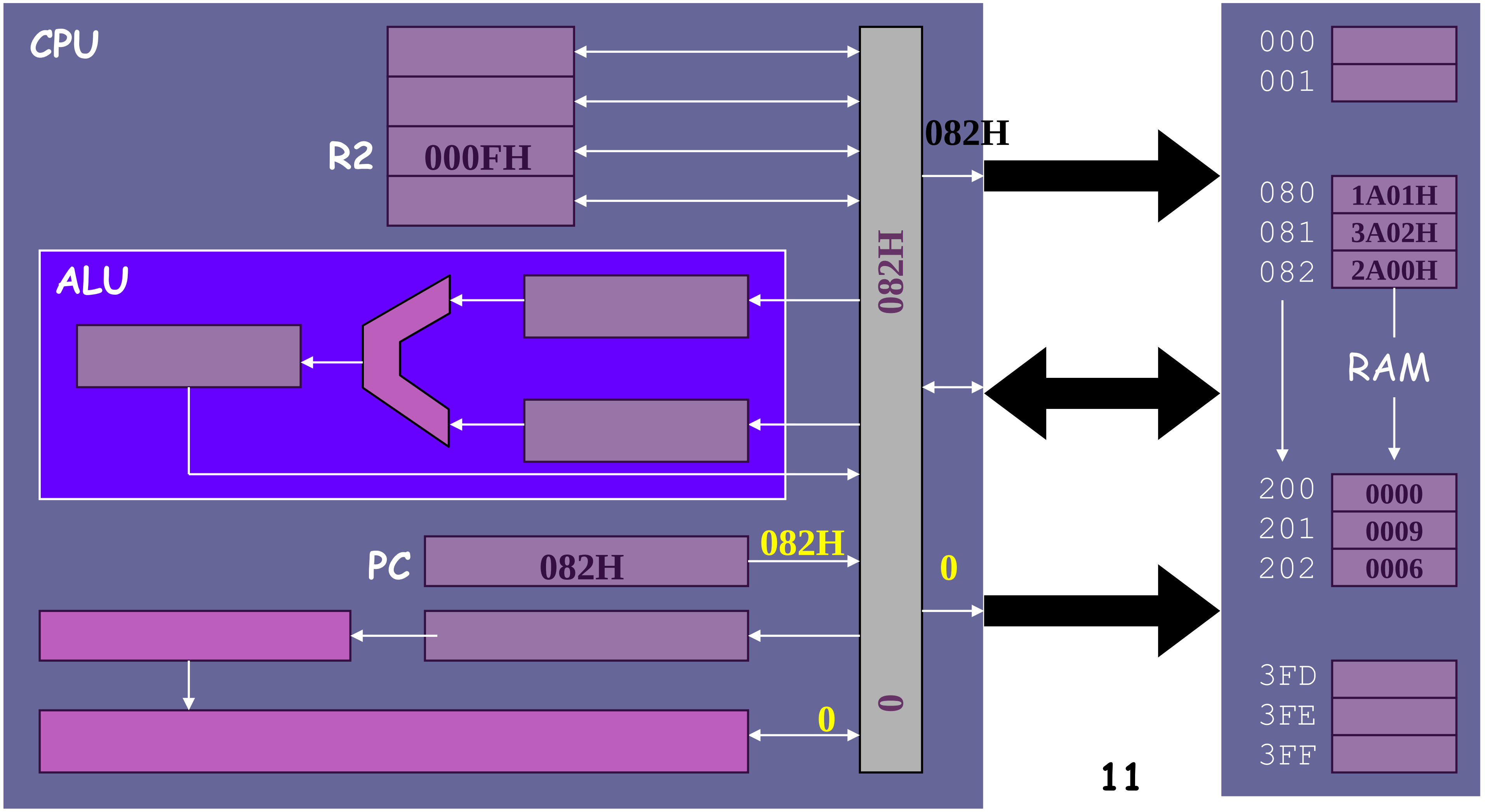
ADD R2, [202H]   R2=R2+Memory[202H]



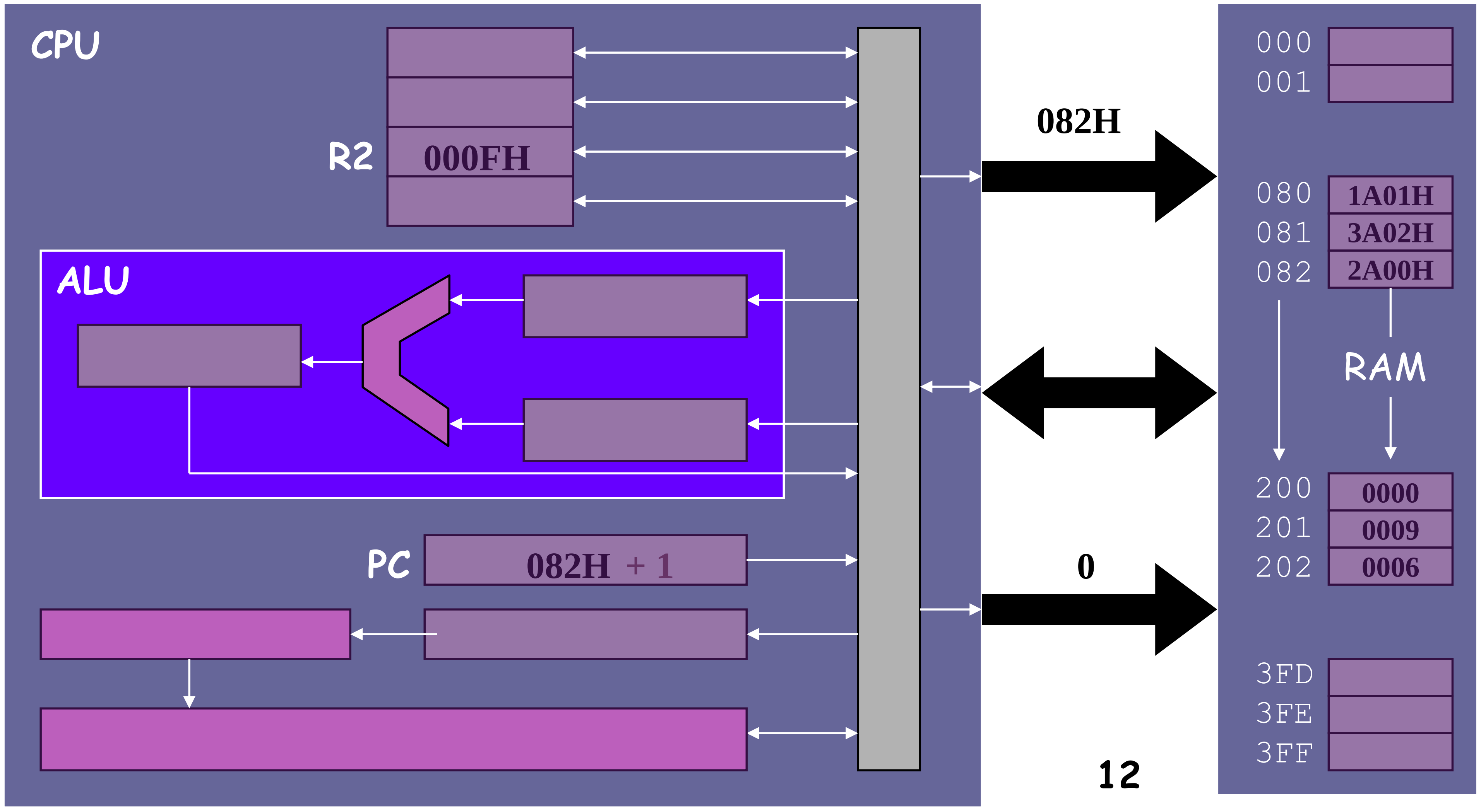
ADD R2, [202H] R2=R2+Memory[202H]



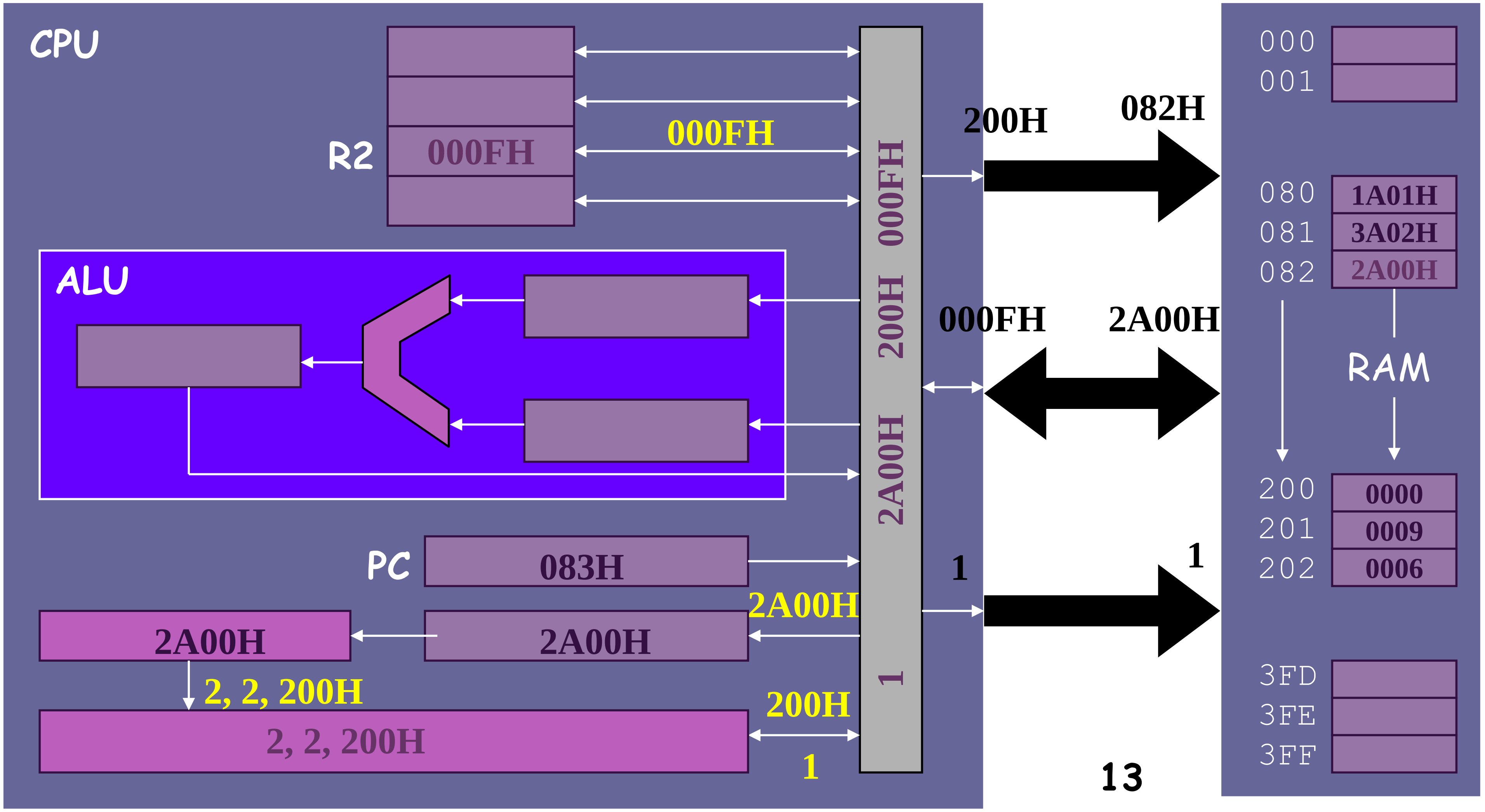
STORE R2, [200H] Memory[200H]=R2



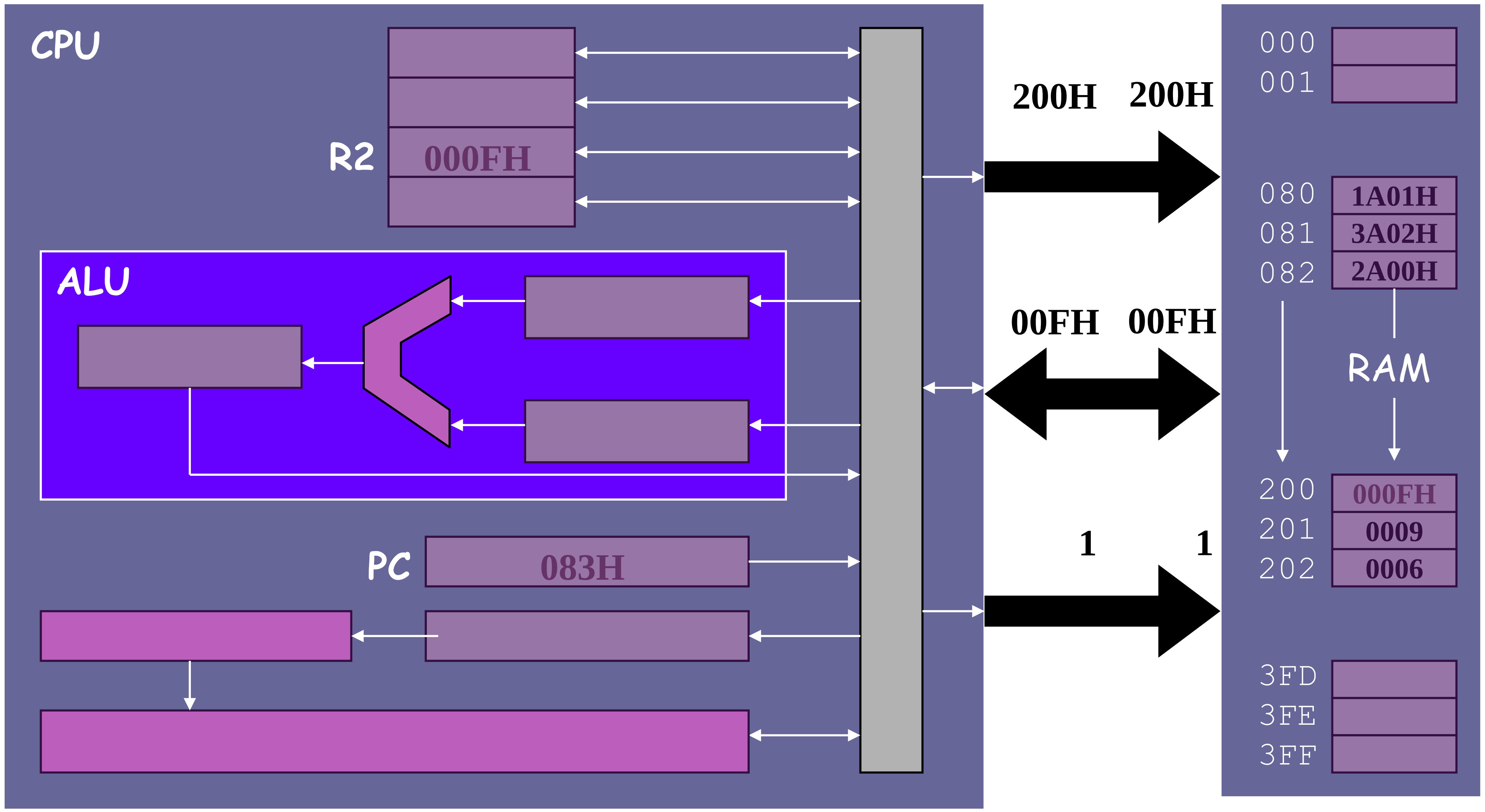
STORE R2, [200H]   Memory[200H]=R2



STORE R2, [200H] Memory[200H]=R2



STORE R2, [200H] Memory[200H]=R2


CPU
R2
000FH
ALU
PC
083H
200H
200H
00FH
00FH
1
1
000
001
080
1A01H
081
3A02H
082
2A00H
RAM
200
000FH
201
0009
202
0006
3FD
3FE
3FF