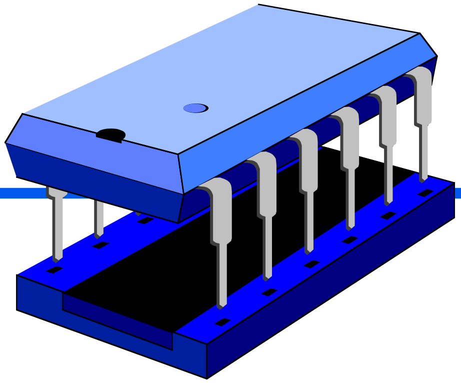


# CPU Organisation & Operation

Professor Kin K. Leung
kin.leung@imperial.ac.uk
www.commsp.ee.ic.ac.uk/~kkleung/

Heavily based on materials by Dr. Naranker Dulay

## Fetch-Execute Cycle

- Fetch the ***Instruction***
- Increment the ***Program Counter***
- Decode the ***Instruction***
- Fetch the ***Operands***
- Perform the ***Operation***
- Store the ***Results***
- **Repeat** Forever

## High-Level/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. Pentium, PowerPC, ARM etc, Java Bytecode)

```
LOAD    R2,  B
ADD     R2,  C
STORE   R2,  A
```

Assembly Language Instructions

---

- (Binary) Machine Code

```
0001101000000001
0011101000000010
0010101000000000
```

Machine Code Instructions

## 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**

---

- Upto **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)

# Other Possibilities for the Format

ADD R2, C

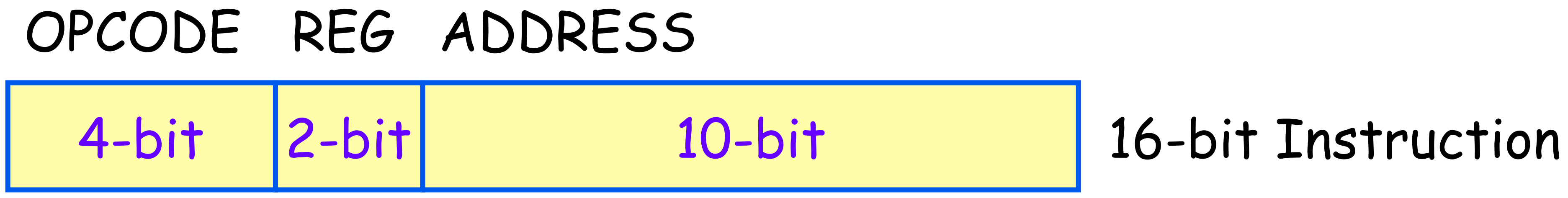
| OPCODE | ADDRESS | REG |
|---|---|---|
| 4-bit | 10-bit | 2-bit |

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

ADD R2, R3

| OPCODE | REG | REG | OPCODE | REG | REG |
|---|---|---|---|---|---|
| 4-bit | 2-bit | 2-bit | 4-bit | 2-bit | 2-bit |

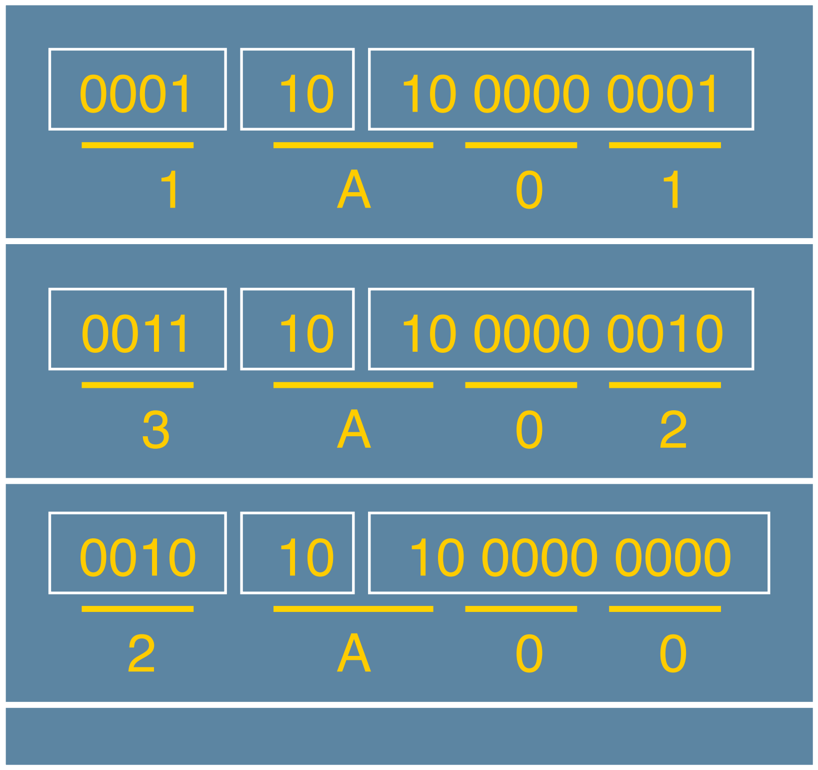
# Instruction Field Encoding



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

# Memory Placement (Program)

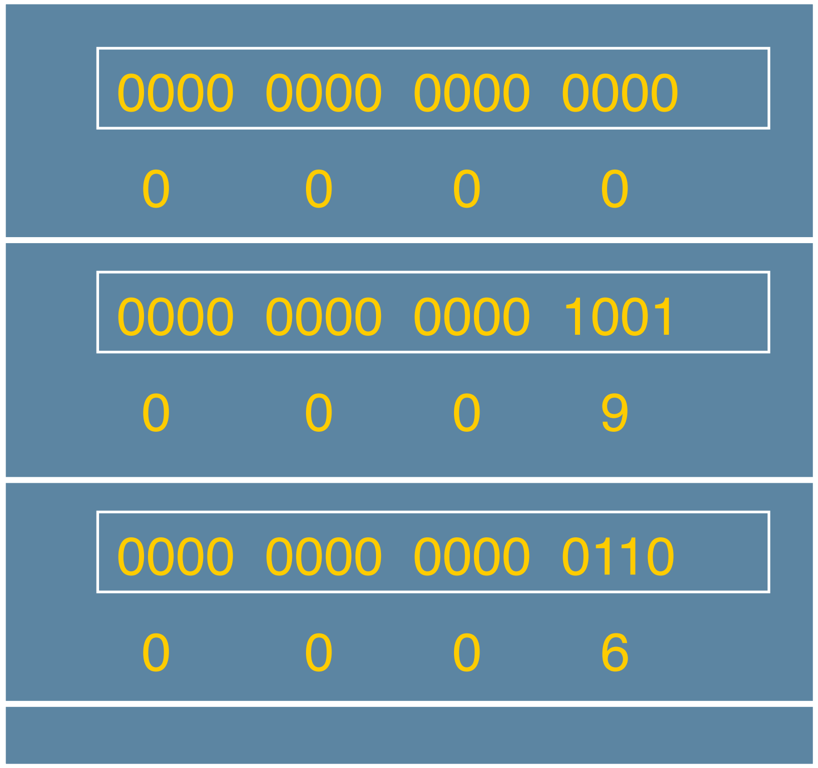
Assembly Instruction | Machine Instruction (OP REG ADDRESS) | Memory Address



| Assembly Instruction | OP | REG | ADDRESS | Hex | Memory Address | Hex |
|---|---|---|---|---|---|---|
| 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

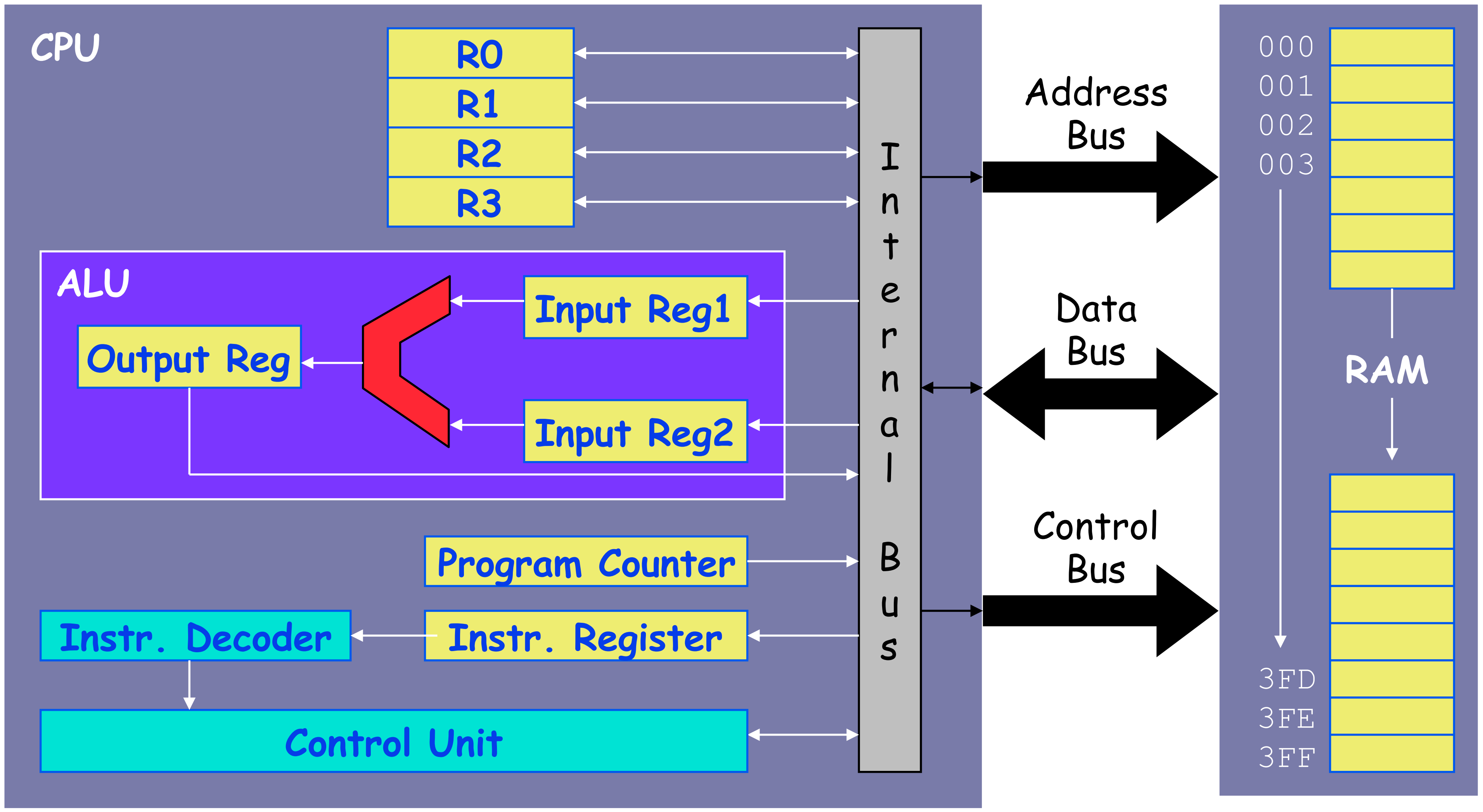
# Memory Placement (Data)



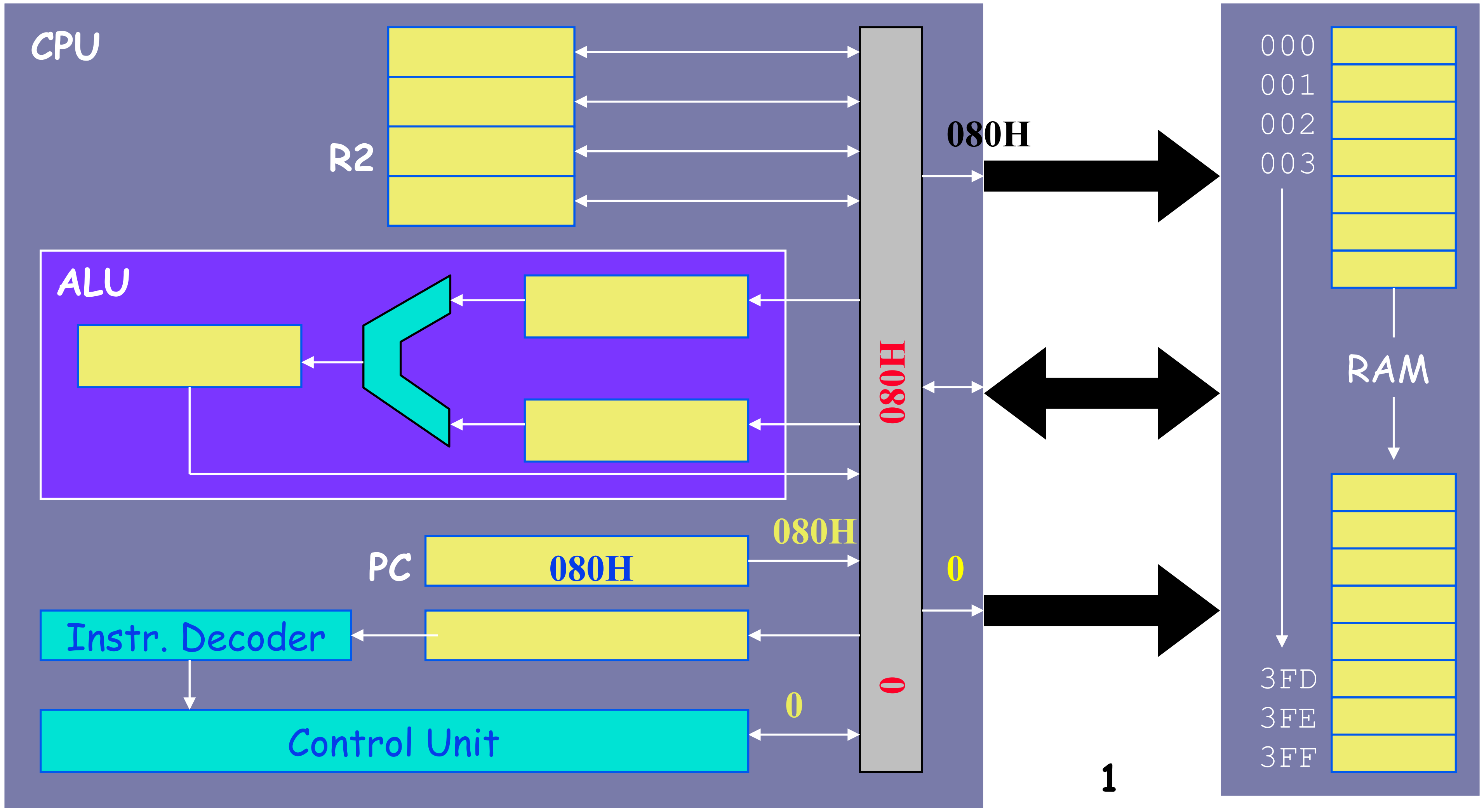
| Assembly Instruction | Data | Memory Address |
|---|---|---|
| A = 0 | 0000 0000 0000 0000<br>0 0 0 0 | 10 0000 0000<br>2 0 0 H |
| B = 9 | 0000 0000 0000 1001<br>0 0 0 9 | 10 0000 0001<br>2 0 1 H |
| C = 6 | 0000 0000 0000 0110<br>0 0 0 6 | 10 0000 0010<br>2 0 2 H |

MEMORY

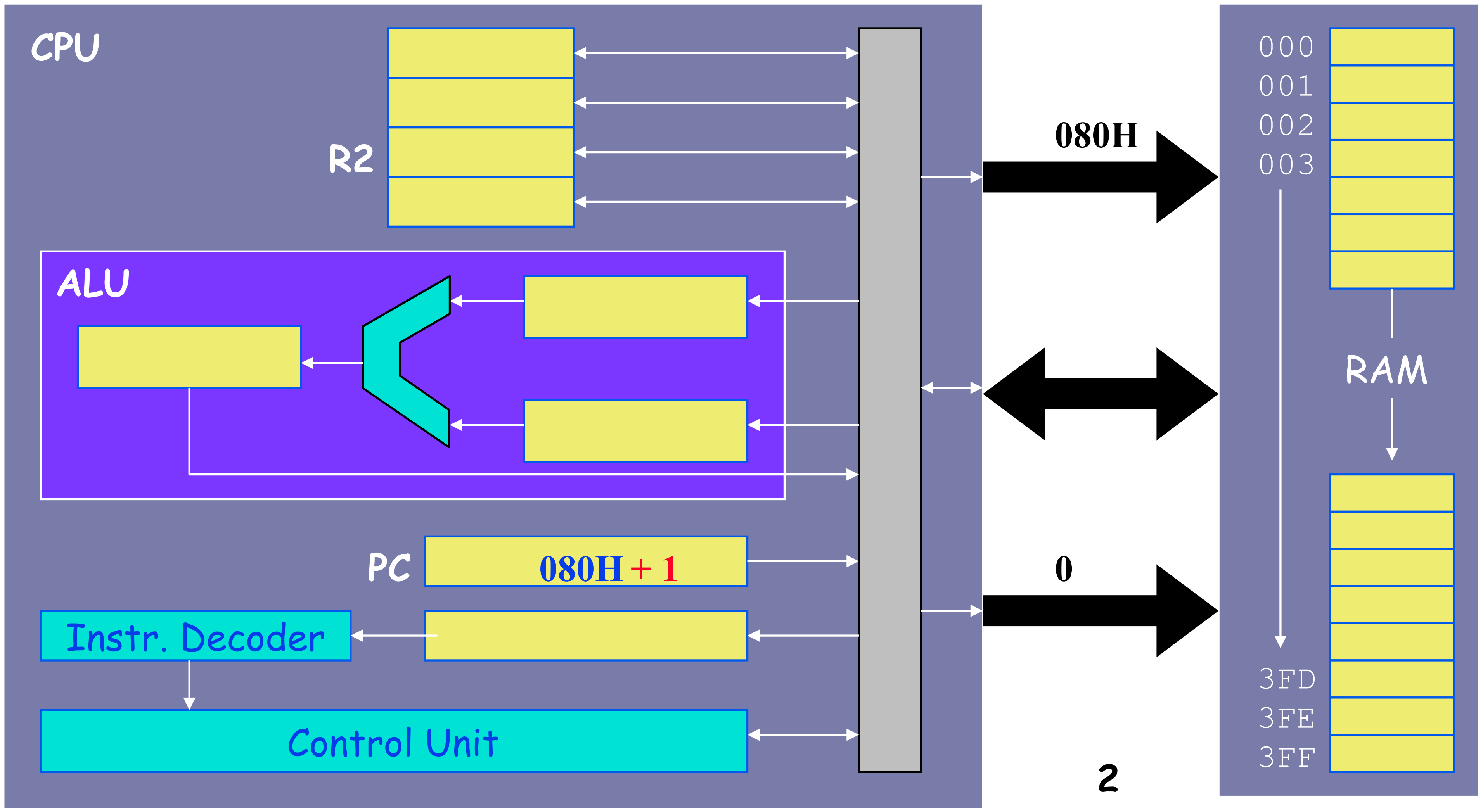
# CPU Organisation



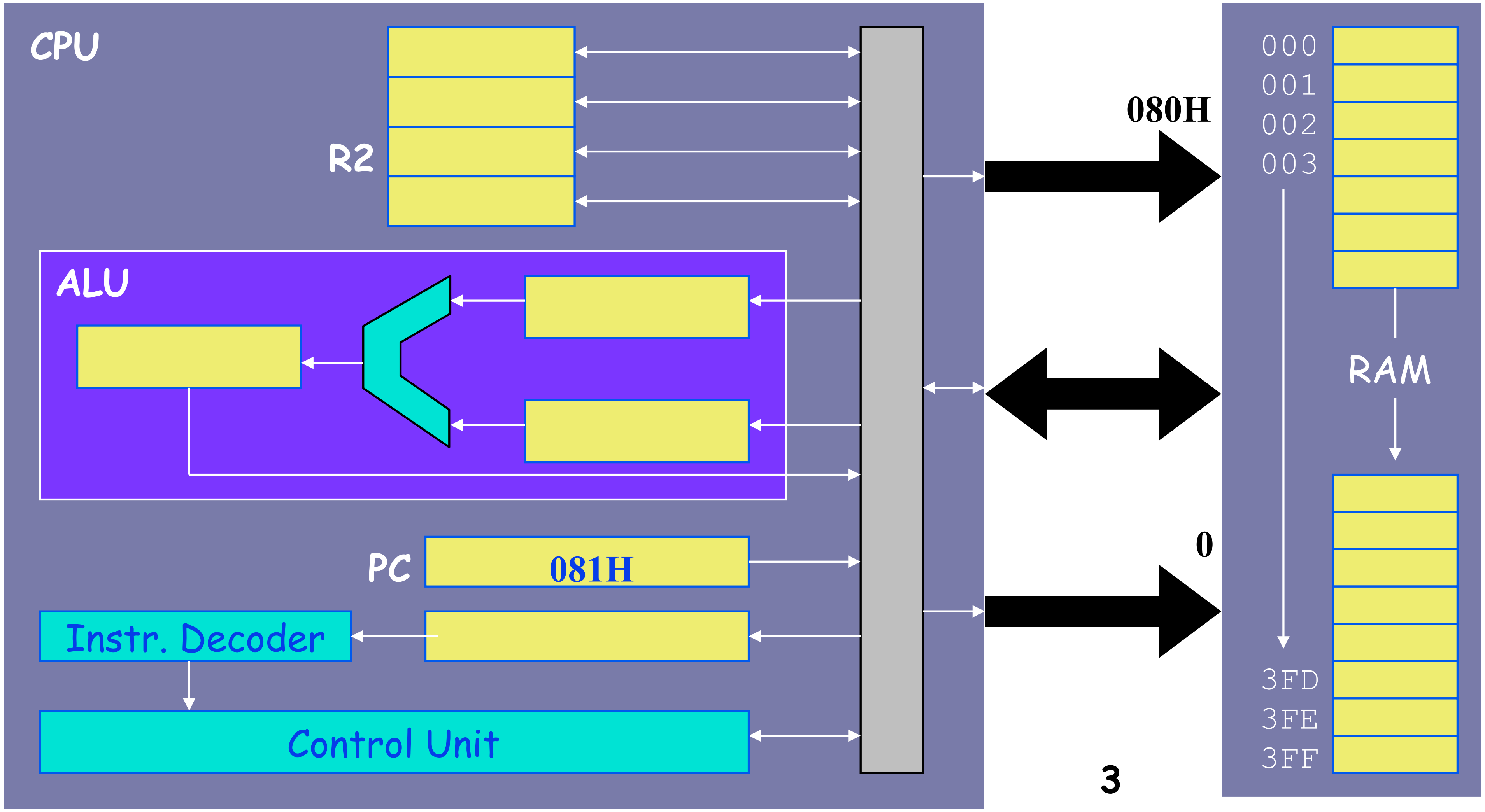
## LOAD R2, [201H]  R2=Memory[201H]



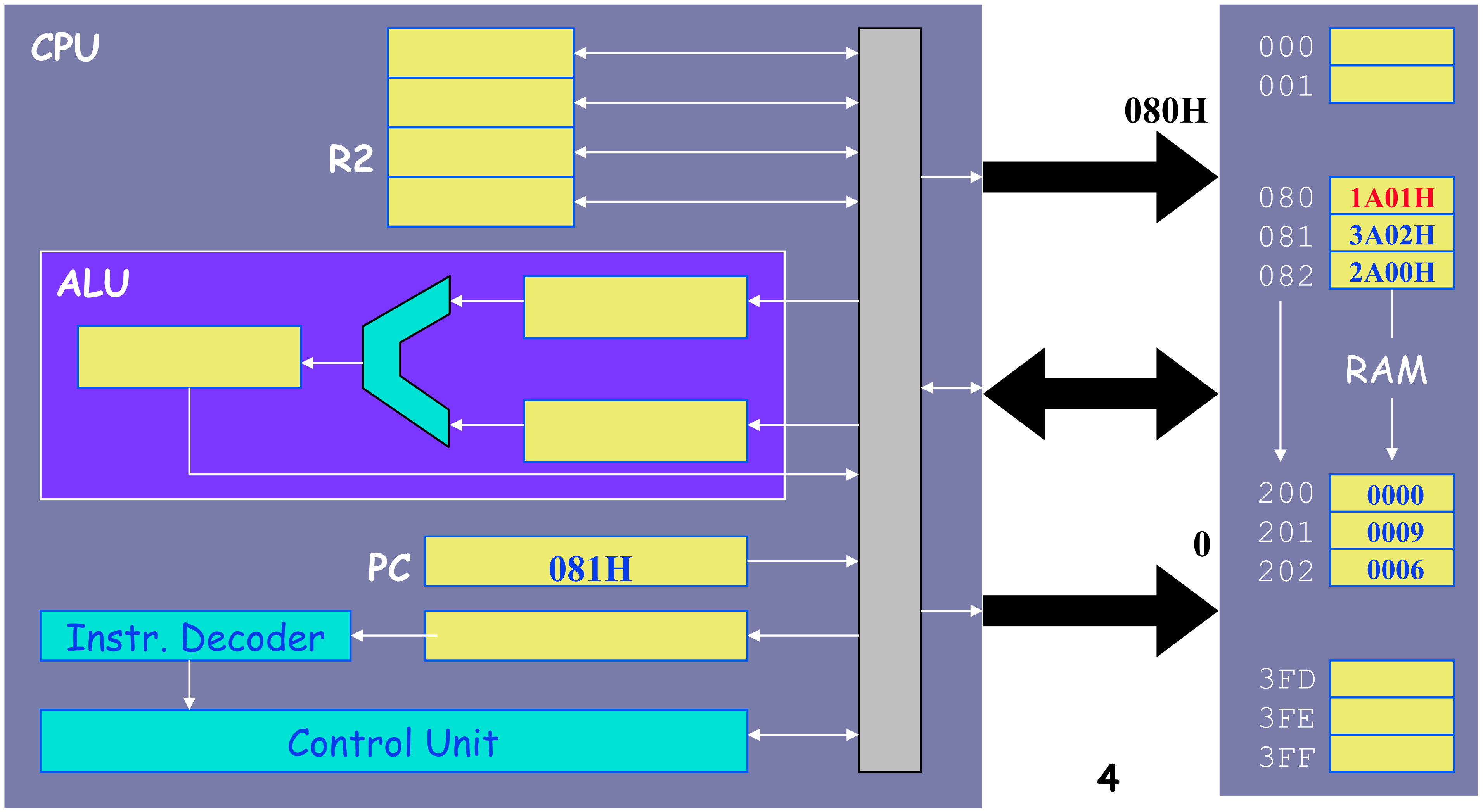
# LOAD R2, [201H]  R2=Memory[201H]



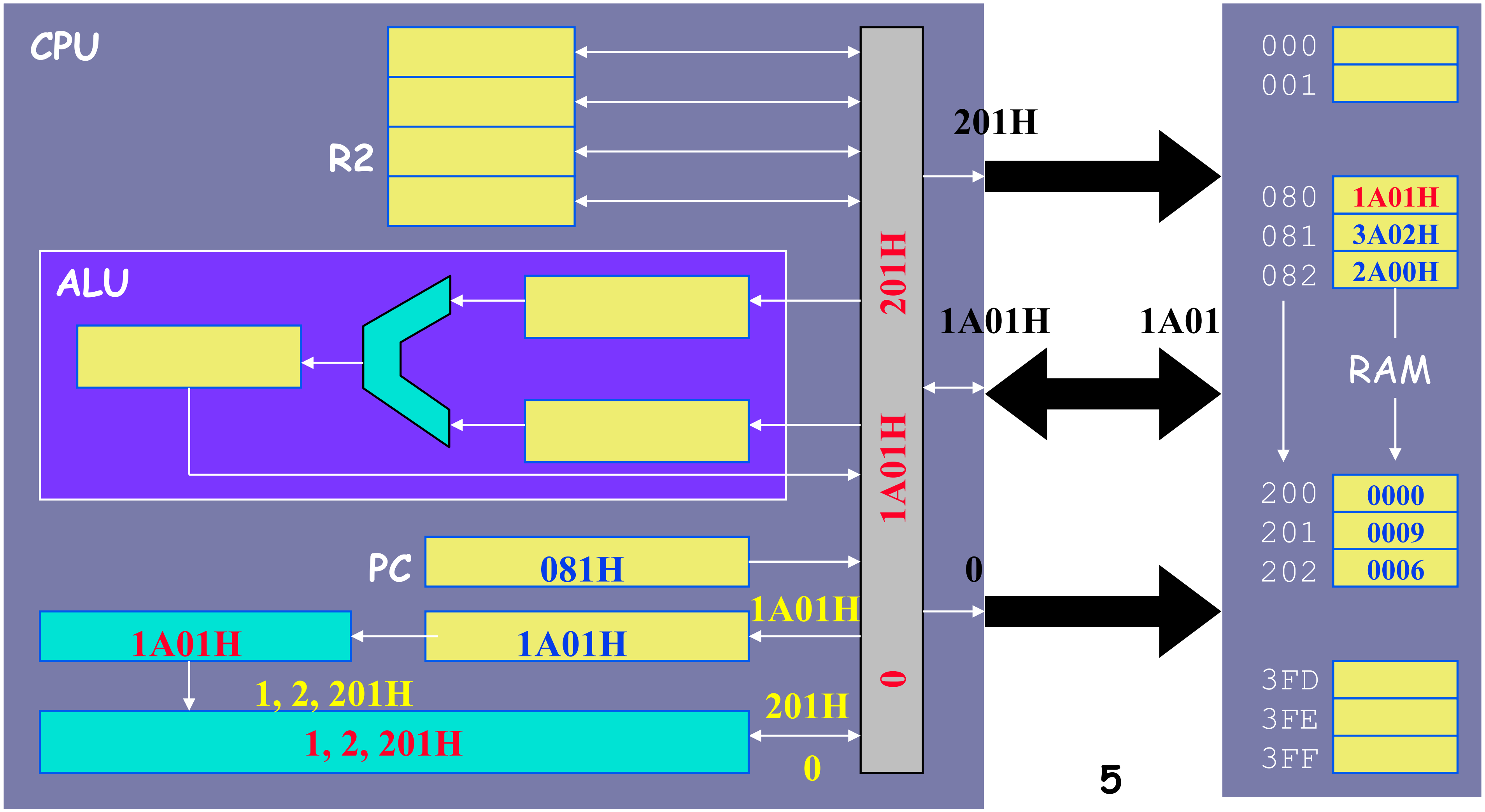
## LOAD R2, [201H] R2=Memory[201H]



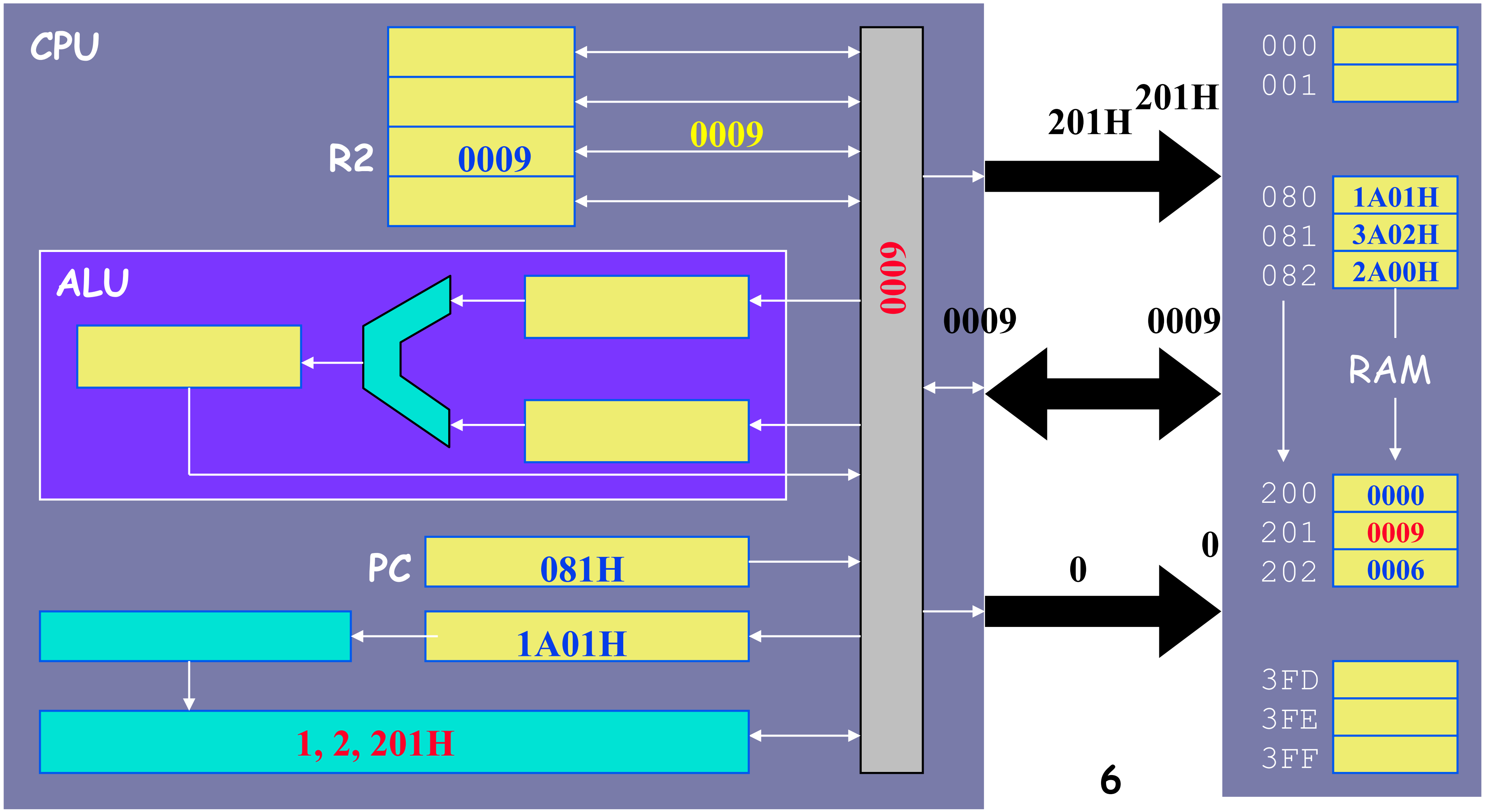
# LOAD R2, [201H] R2=Memory[201H]



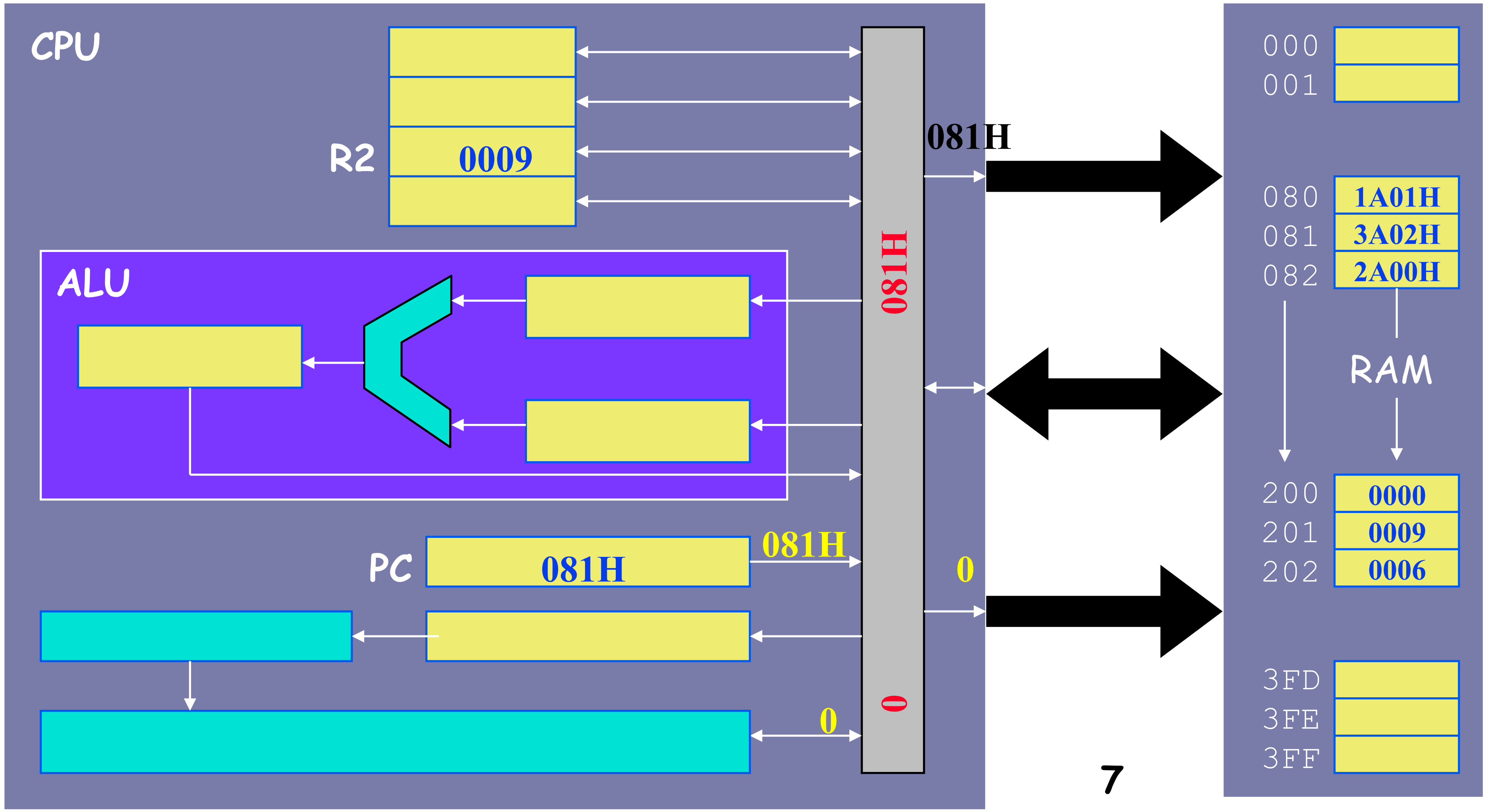
# LOAD R2, [201H]    R2=Memory[201H]



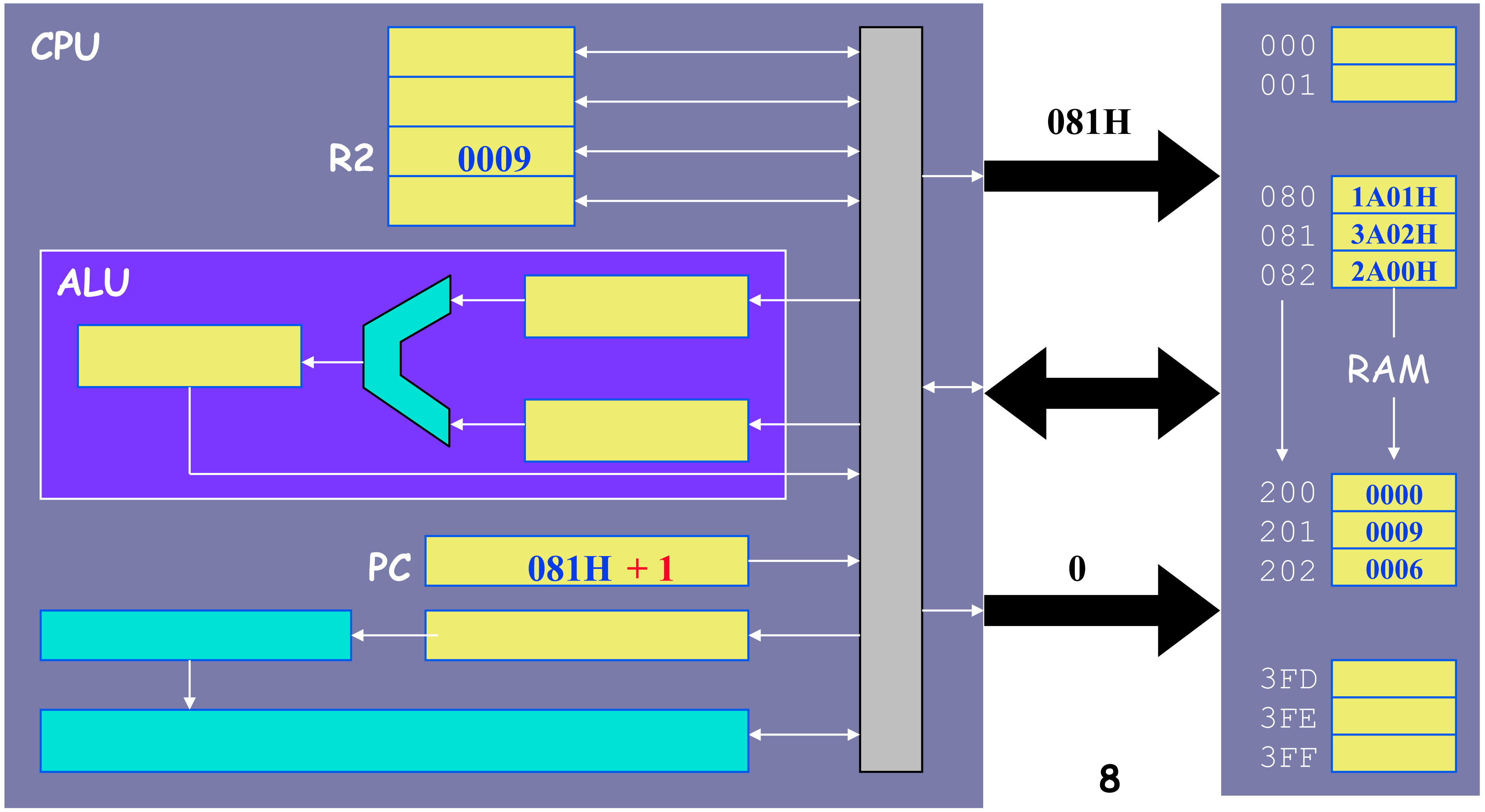
# LOAD R2, [201H]      R2=Memory[201H]



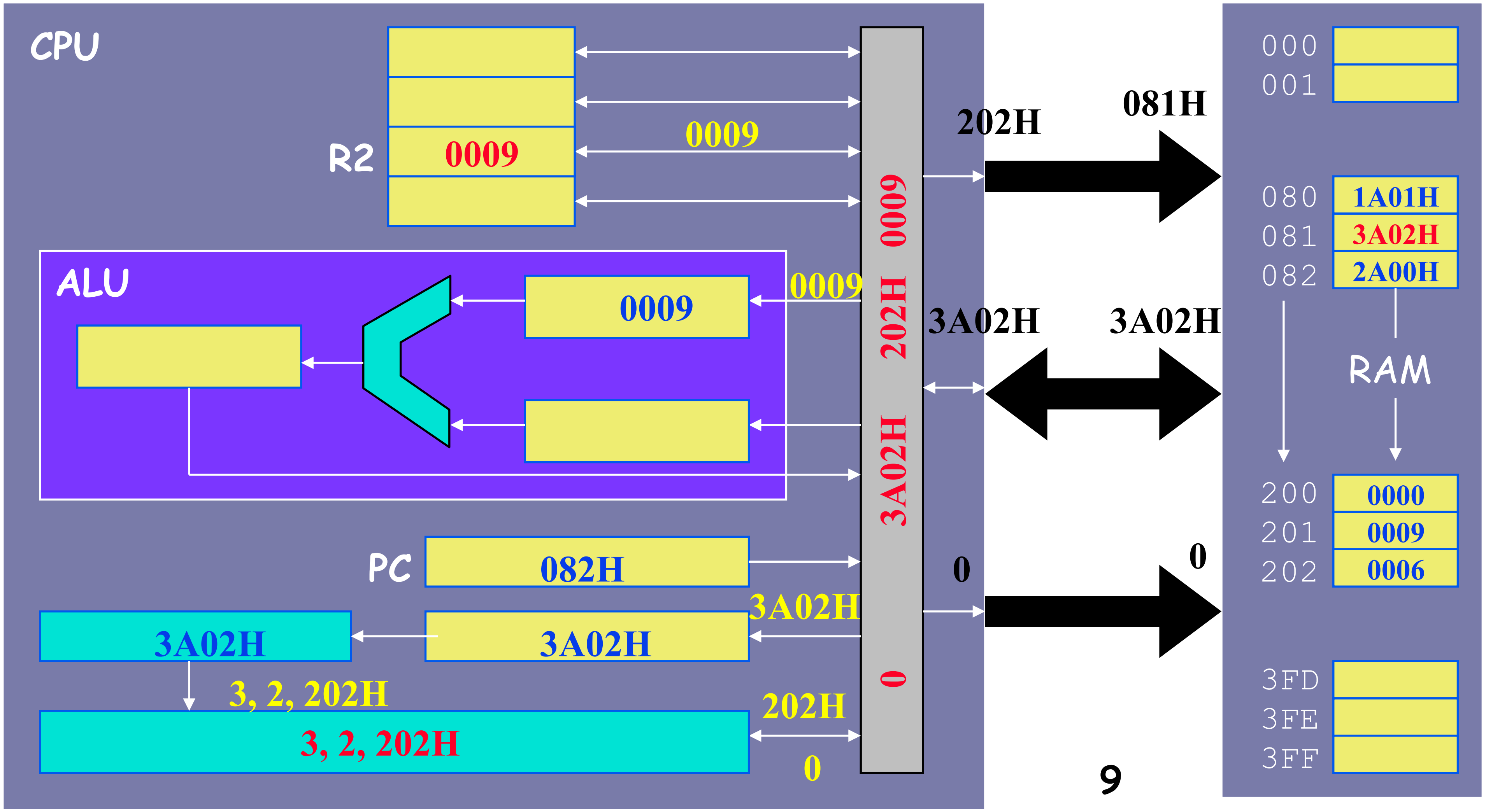
## ADD R2, [202H]  R2=R2+Memory[202H]



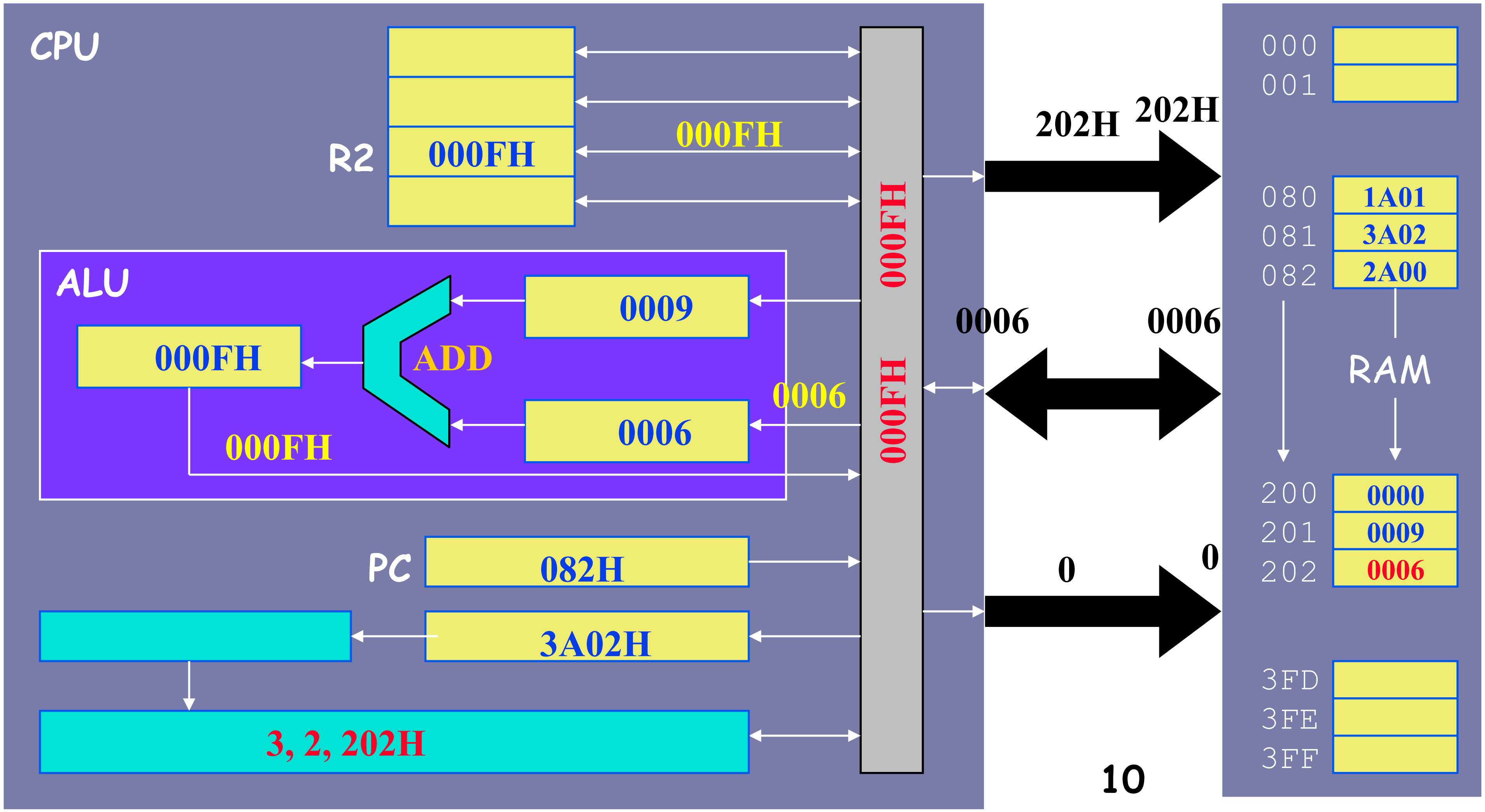
## ADD R2, [202H]   R2=R2+Memory[202H]



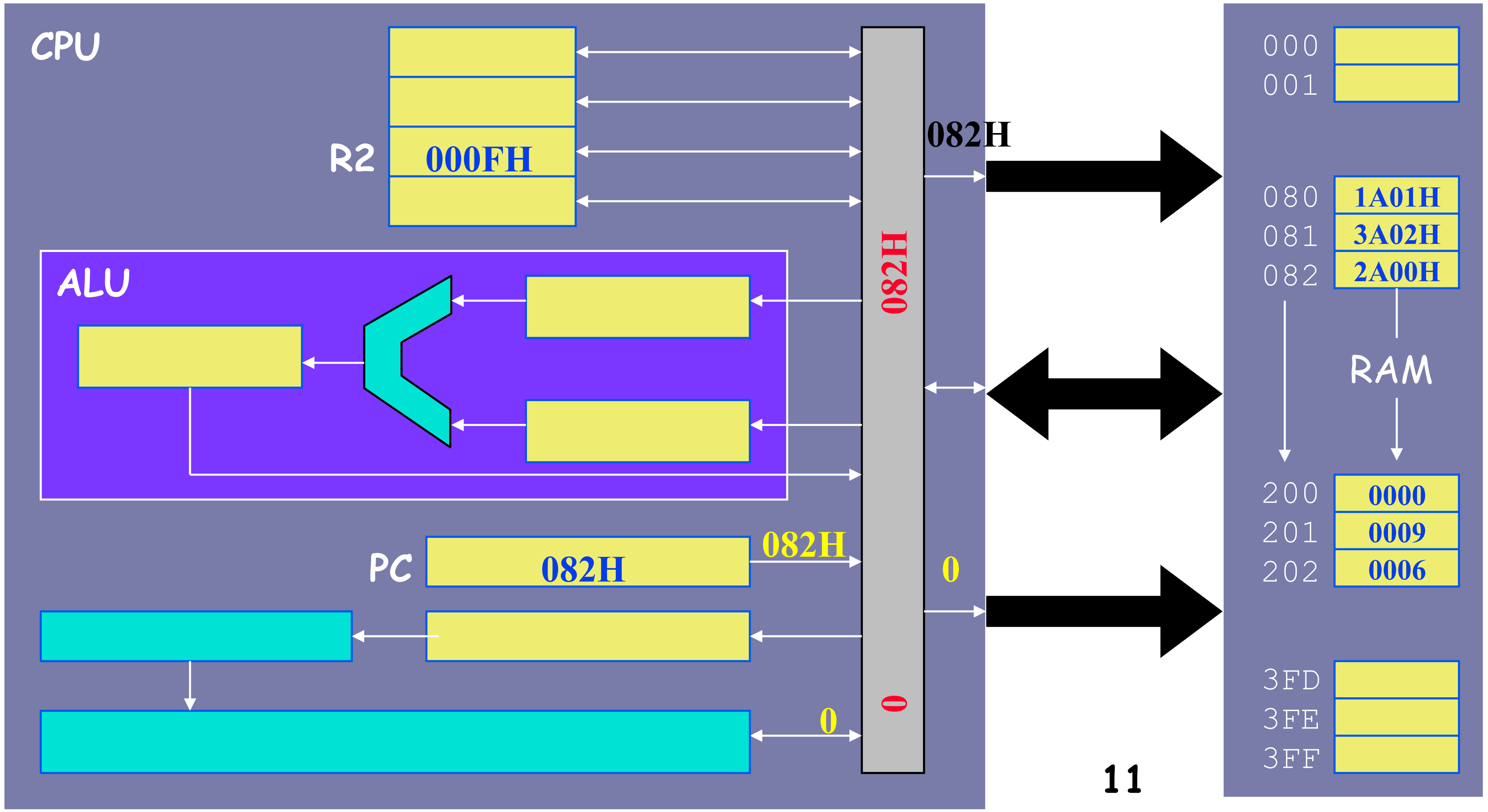
## ADD R2, [202H]    R2=R2+Memory[202H]



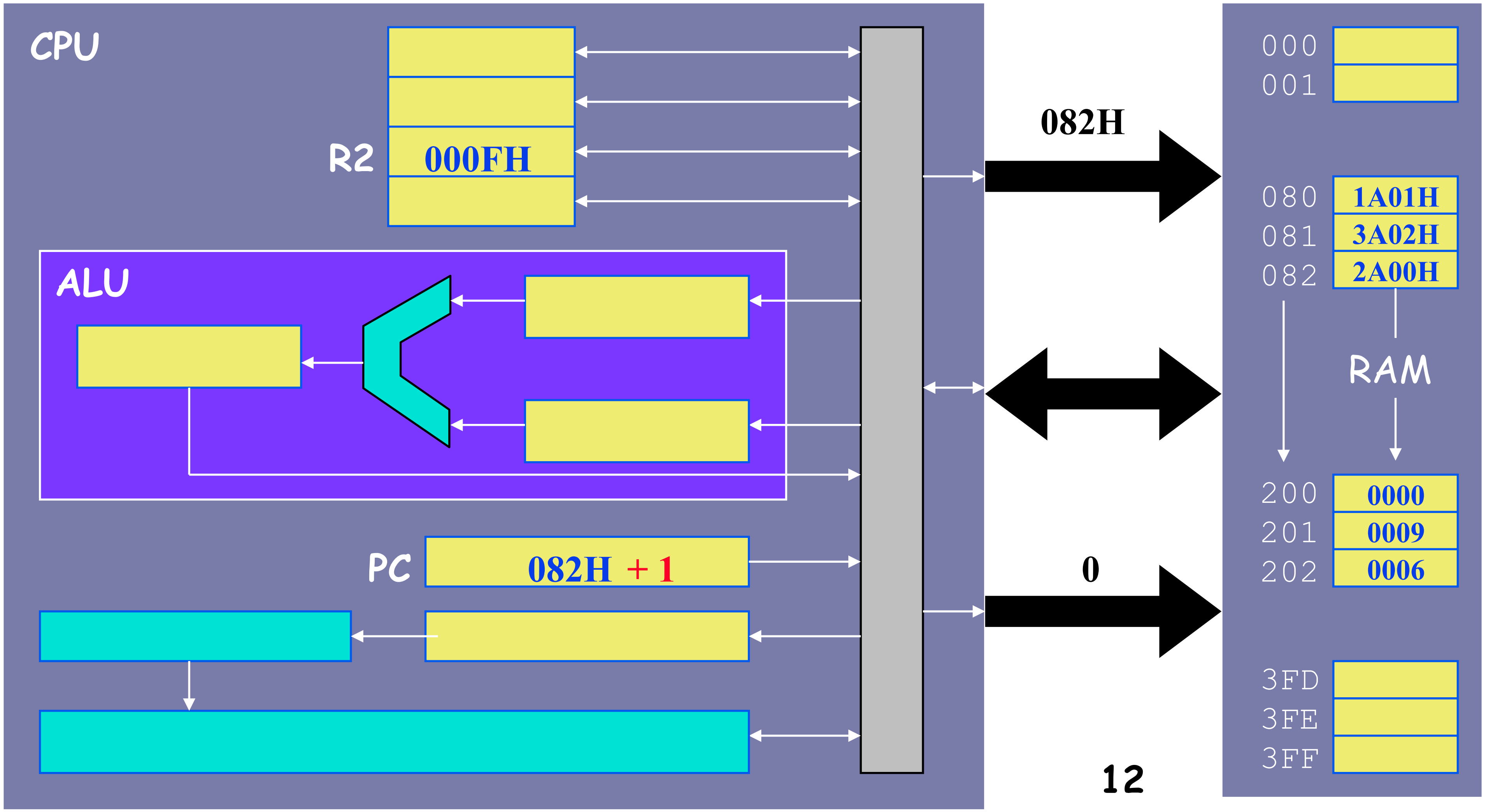
## ADD R2, [202H]  R2=R2+Memory[202H]



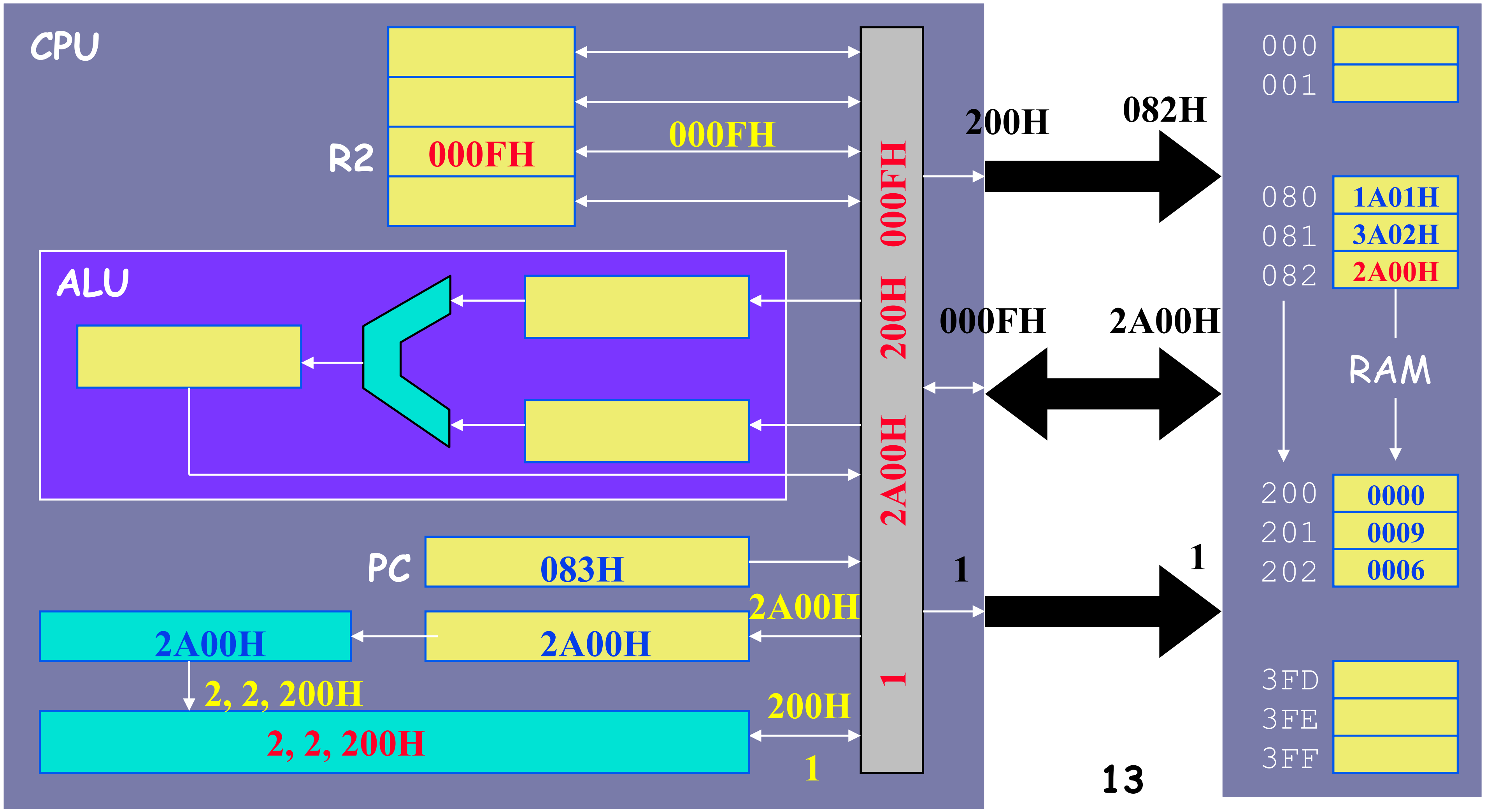
## STORE R2, [200H]    Memory[200H]=R2



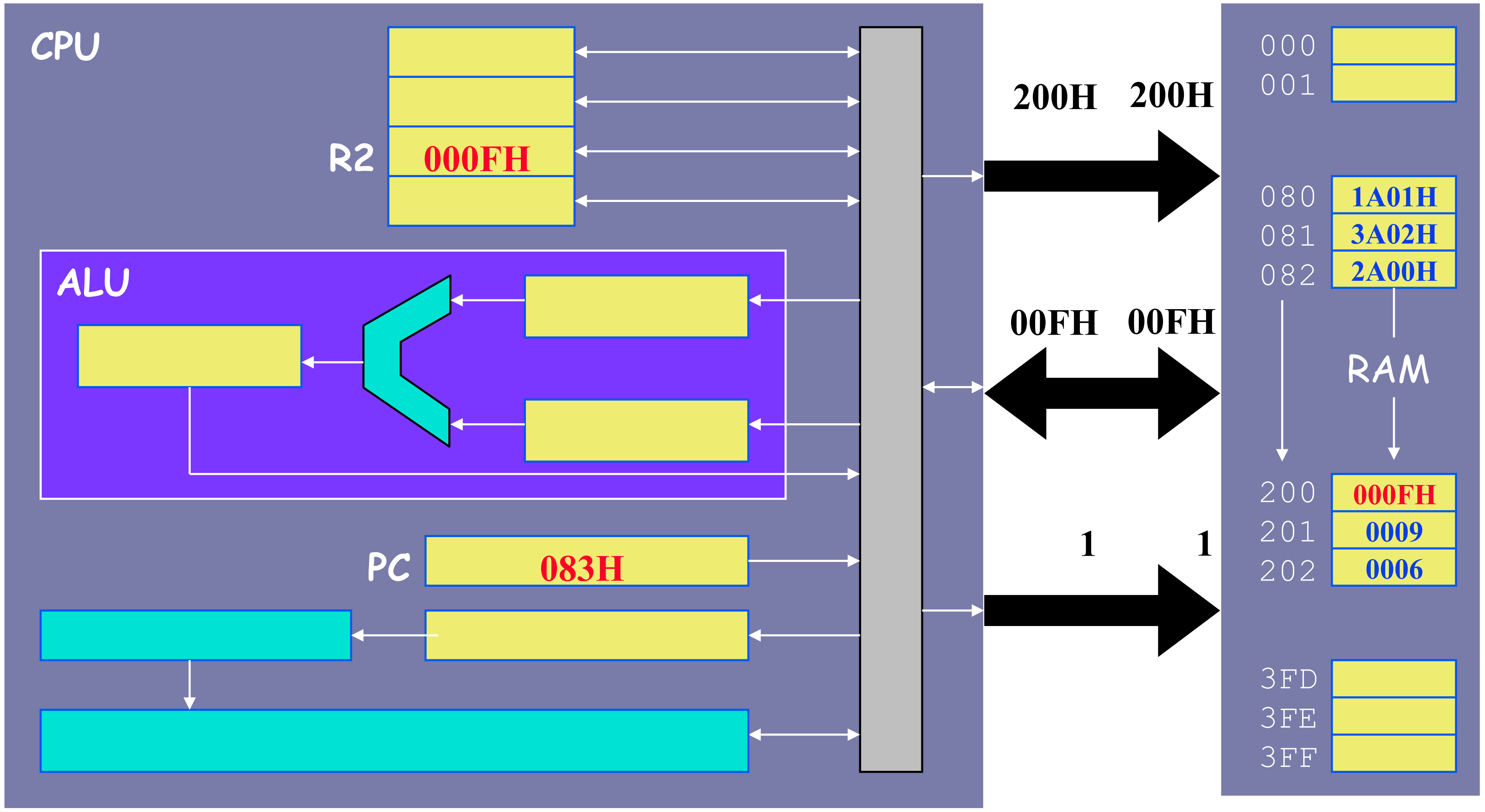
## STORE R2, [200H]　　Memory[200H]=R2



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



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



# Think About

| | |
|---|---|
| ➢ Fetch-Execute Cycle | |
| ➢ Assembly Languages | |
| ➢ Program Representation: | Instructions, Instruction Fields, Instruction Formats |
| ➢ CPU Components: | Registers, ALU, Control Unit |
| ➢ Registers: | General Purpose Registers, Program Counter (PC), Instruction Register (IR), ALU Registers |
| ➢ Buses: | Internal, Address, Data, Control |

---

- ➢ Next Topic

  TOY1 Assembly Programming