# Lecture 5

Addressing Methods



## Topics

- Addressing Methods of instruction operands

# Register and Status Flag Names in Educational CPU

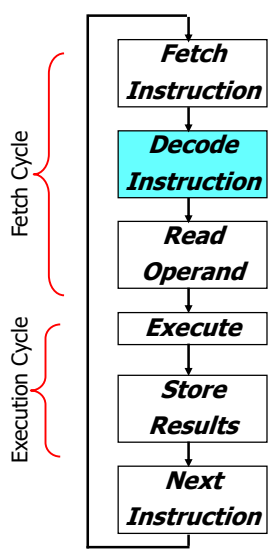
| 8 Bit Registers | |
|---|---|
| Accumulator | A |
| Accumulator | B |
| Auxilary register | C |
| Auxilary register | D |
| Status Flags register (Durum Kütüğü) (Condition Codes Register) | DK |

| 16 Bit Registers | |
|---|---|
| Accumulator pair | AB |
| Auxilary register pair | CD |
| Index Register (IX) (Sıralama Kütüğü) | SK |
| Stack Pointer (SP) (Yığın Göstergesi) | YG |

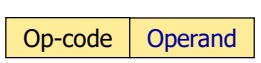
| | Status Flag Bit Names in Condition Codes Register | | | | | |
|---|---|---|---|---|---|---|
| | Interrupt (Kesme) | Overflow (Taşma) | Zero (Sıfır) | Negative (Negatif) | Half Carry (Yarım Elde) | Carry (Elde) |
| English | I | V | Z | N | H | C |
| Turkish | K | T | S | N | Y | E |

3

# Detecting the Addressing Mode of an Instruction

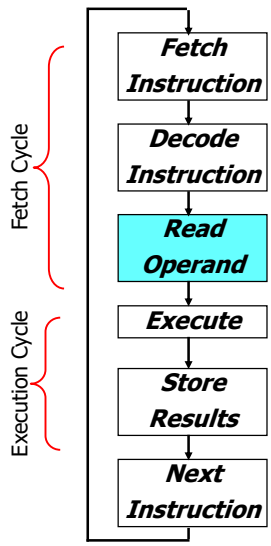



- During the instruction decoding phase, the **addressing mode** is detected and the Effective Address is obtained by CPU.
- Addressing mode identifies where to find the **operand** in the memory or among the registers.




4

# Possible Sources of Operands




- During the read operand phase, the **operand value** is obtained by CPU.
- Operand can be one of the followings.
  - Memory location
  - Registers
  - Immediate Data

5

# Addressing Methods

- **Major Addressing Methods**
  1. Immediate (İvedi)
  2. Register (Doğal)
  3. Direct (Doğrudan)
  4. Indirect (Dolaylı)
  5. Indexed (Sıralı)
  6. Relative (Bağıl)
- **Advanced Addressing Methods**
  1. Memory Immediate Write (İvedi Yaz)
  2. Incremented Index (Artırmalı Sıralı)
  3. Decremented Index (Azaltmalı Sıralı)
  4. Register Relative Index (Kütüğe Bağlı Sıralı)
  5. Stack Pointer Relative (Yığın)

6

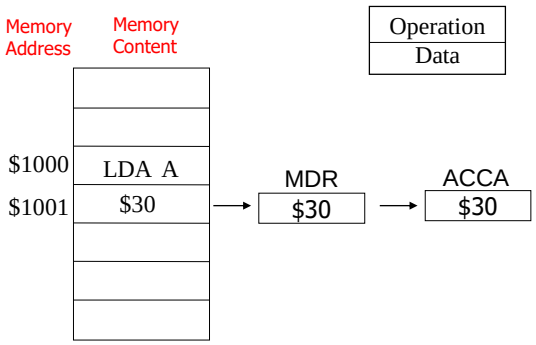
# 1- Immediate Addressing

- The operand (immediate data) is contained in the instruction.
- Instruction does not specify a memory address location.

Example:

**LDA A, $30**

- Load hexadecimal 30 (immediate data) to accumulator register A.
- $ symbol is the prefix for hexadecimal numbers.




- Opcode for the "LDA A" instruction is actually 2 bytes in length.
- For simplicity, here it is assumed as 1 byte instruction stored in address $1000.

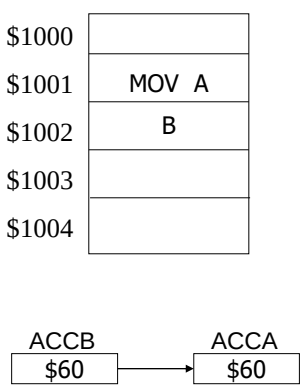
7

# 2- Register Addressing

- The instruction contains the **register names only**.
- No addressing to the memory.
- Short instruction length (1 or 2 Bytes)

**MOV A, B**

- Move (copy) content of accumulator B to accumulator A.



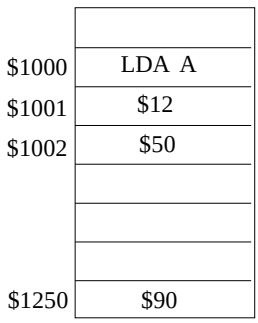

8

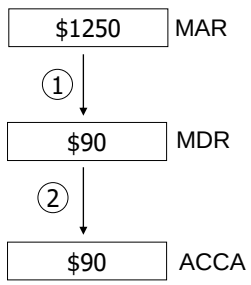
# 3- Direct Addressing

- Instruction contains the memory address of the operand.
- Effective address of the operand is in the instruction.

**LDA A, <$1250>**

- Load data from address $1250 to accumulator A.
- Angle paranthesis symbols < > means memory address.



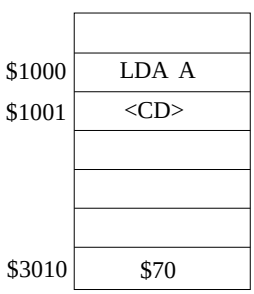

# 4- Indirect Addressing

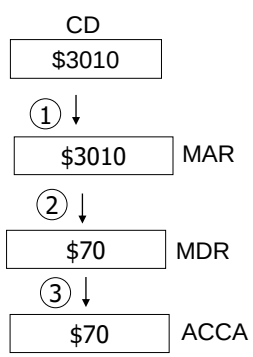
- Instruction contains the CD register as operand.
- **The CD register pair is used for indirection.**
- Effective address is at the address location specified in the CD register.

*Store data $70 at address
STA $70, $3010

*Load address to CD register
LDA CD, $3010

*Load data from memory to accumulator A
**LDA A, <CD>**



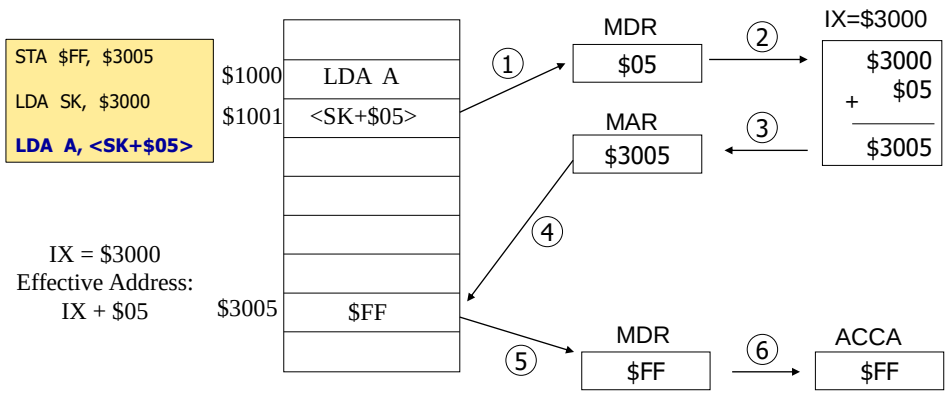




**LDA CD, $3010** instruction can be re-written with two steps as follows:

```
LDA  C,  $30  ;high byte of address
LDA  D,  $10  ;low byte of address
```

# 5- Indexed Addressing

- **Syntax :** <IX+Offset>
- Index Register (IX) / Sıralama Kütüğü (**SK**) is 2 bytes.
- It can be used as an index on an array, and also as a loop counter.
- Another usage may be as a temporary storage for a two-byte data.
- The operand's effective address = IX (2 byte) + Constant offset number (1 or 2 byte)




## Example1: Using SK register as source operand

- SK register (Index Register IX) usage as a source operand requires an additional **constant number** (as base or as offset).
- If there is no additional constant number, then **zero** must be written.

**LDA A, <SK>** → Compiler error: "Unresolved symbol: SK"

**LDA A, <SK+0>** → Correct syntax

12

## Example2: Using SK register as target operand

- SK register can be used as target operand (destination).
- It can be assigned with two-byte value (either with immediate data or with memory address).

**LDA SK, $2000** → Load SK with immediate value $2000 ($2000 can be a data or an absolute memory address)

**LDA SK, <$2000>** → Load SK with content of memory address $2000 and $2001

**LDA SK, ARRAY** → Load SK with memory address of variable that is named ARRAY

13

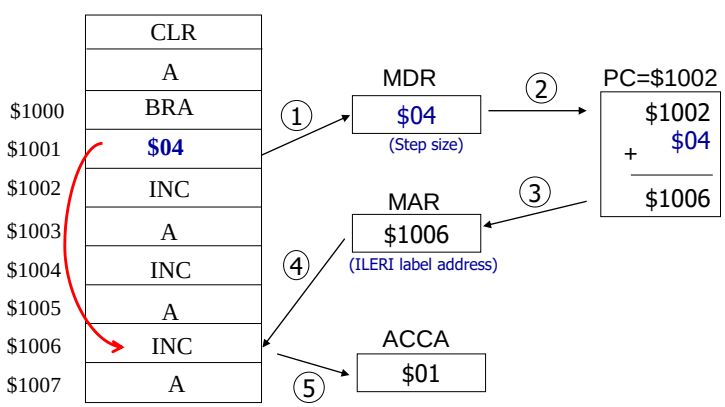
## 6- Relative Addressing

- Used for **branching** instructions only.
- Not used for data transfers from/to memory.
- Branch address is relative to the Program Counter (PC).
- Effective Address = PC +/- [Step size] (step is 7 bit : ± 128)
- Step size (number of offset bytes) is computed by compiler.

Example: Unconditional branch (goto) to a label.

BRA Label

```
  CLR  A
  BRA  ILERI
  INC A
  INC A
ILERI
  INC A
```




Wrong usage:
BRA $04

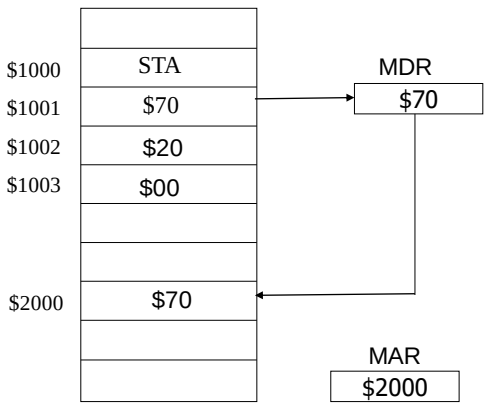
Branches to absolute address $0004 !

# Advanced Addressing Methods

- **1- Memory Immediate Write Addressing**
  - Immediate data is written directly to a memory location

```
STA  $70, $2000
```

- Store data $70 to address $2000.



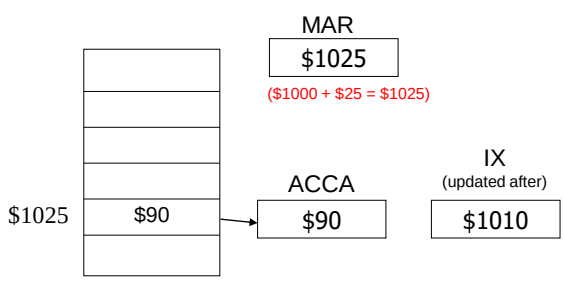



# Advanced Addressing Methods

- **2- Incremented Index Addressing**
  - **Syntax :** <IX+Offset> + Range
  - Effective address of operand is computed as IX + Offset, by CPU.
    - Firstly, the load operation is done from <IX+Offset>
    - Then, IX is incremented by the Range amount, provided at the instruction.
  - Length of Range is 1 Byte ($00 to $FF).

```
STA  $90, $1025
LDA  SK, $1000
LDA  A, <SK+$25> + $10
```

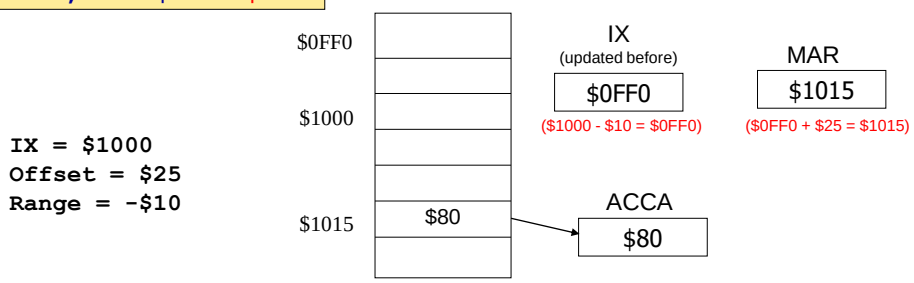
```
IX = $1000
Offset = $25
Range = $10
```



# Advanced Addressing Methods

- **3- Decremented Index Addressing**
  - **Syntax :** <IX+Offset> - Range
  - First, the IX is decremented by the Range (before the operation).
  - Then, Effective Address of operand is found = IX+Offset

```
STA  $80, $1015
LDA  SK, $1000
LDA  A, <SK+$25> - $10
```

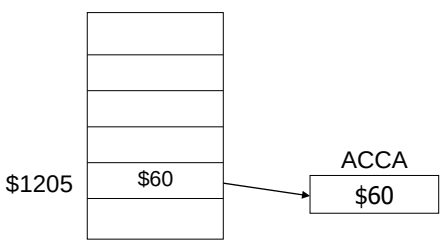



17

# Advanced Addressing Methods

- **4- Register Relative Index Addressing**
  - **Syntax :** <CD+IX+Offset>
  - Effective address computed as the sum of General Purpose Registers (**CD**), Index Register (**IX**), and the Offset provided in the instruction.
  - Size of CD and IX registers are the same (16 bits).

```
STA  $60, $1205
LDA  SK, $1000
LDA  CD, $0200
LDA  A, <CD+SK+$05>
```




CD: $0200

IX: $1000

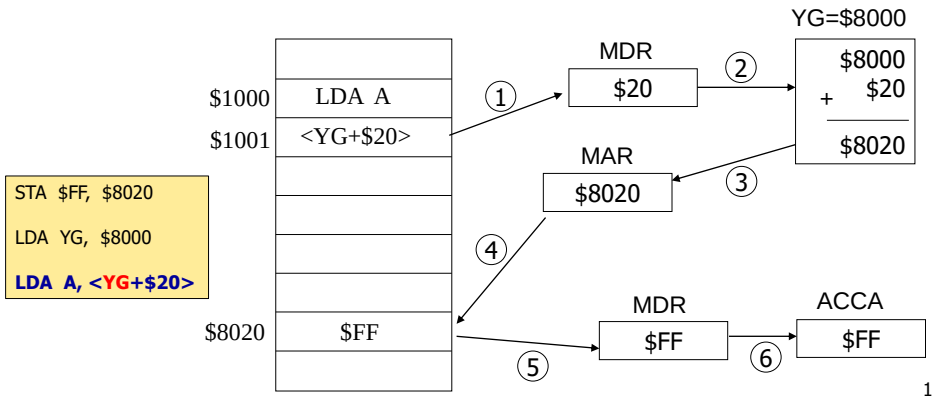
MAR: $1205

($0200 + $1000 + $05 = $1205)

18

# Advanced Addressing Methods

- **5- Stack Pointer Relative Addressing**
  - **Syntax :** <SP+Offset>
  - This method is very similar to using the Index Register (IX).
  - Instead of IX register, the Stack Pointer (**SP**) can be used as an index.
  - **Restriction:** It can be used only if the Stack is not being used for another purpose in the program.




19