MICROPROCESSOR SYSTEMS

Lecture 4

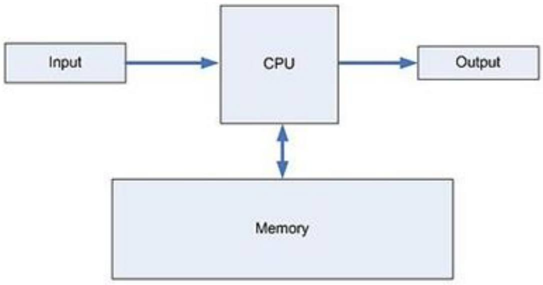
CPU Structure,
Instruction Format

Topics

- Central Processing Unit Structure
- Instruction Format

Central Processing Unit

- CPU is the fundamental execution/processing unit of the computer.
- CPU consists of ALU, Control Unit, and Registers.



CPU Elements

- **Control Unit**
- **Arithmetic and Logic Unit (ALU)**
- **Instruction Decoder**

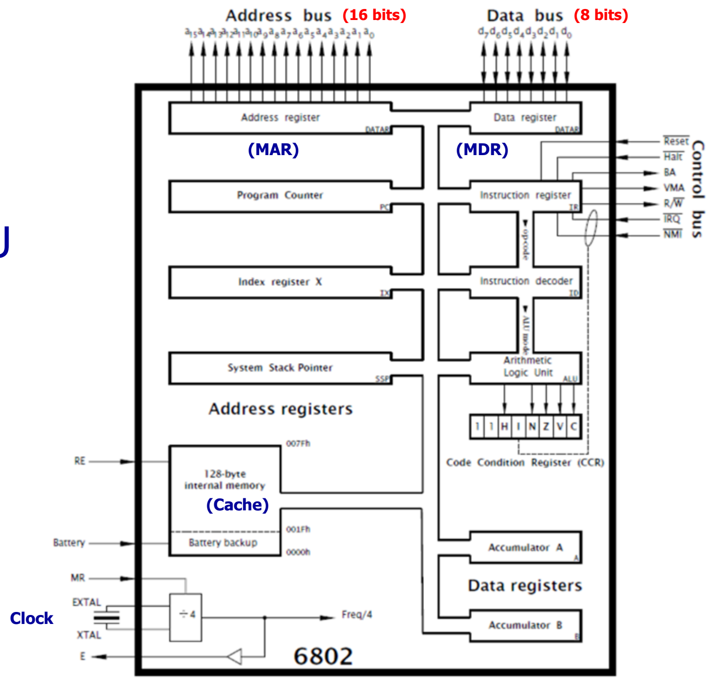
- **ALU Related Data Registers:**
  - Accumulator (**A, B**)
  - General Purpose Registers (**C, D,** …)
  - Condition Code Register (**CCR**)

- **Memory Related Registers:**
  - Memory Address Register (**MAR**)
  - Memory Data Register (**MDR**)
  - Program Counter (**PC**)
  - Instruction Register (**IR**)
  - Stack Register (Pointer) (**SP**)
  - Index Register (**IX**)

Registers shown in red color can be used by programmer in an Assembly program.

Example:

Motorola 6802 CPU (8-bits)

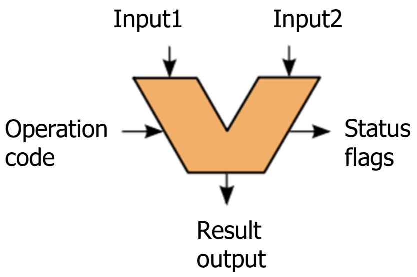


CPU Components

- **Memory Address Register (MAR)** stores the memory address from which data will be fetched to the CPU, or the address to which data will be sent and stored.
- **Memory Data Register (MDR)** contains the data to be stored in the memory, or the data after a fetch from the memory.
- **Accumulator (ACC)** may contain data to be used in a arithmetic or logical operation, or it may contain the result of an operation.
- **General purpose registers** are used to support the accumulator by holding data to be loaded to/from the accumulator.
- **Arithmetic and Logic Unit** (**ALU**) performs all arithmetic and logic operations in a microprocessor.

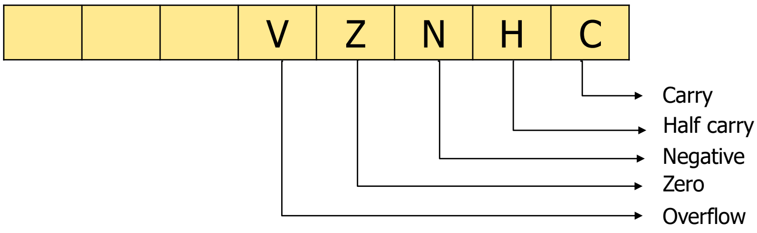
Arithmetic Logic Unit

- Operation code : Control signal that selects the operation
- Input1 , Input2 : Input operands
- Result output : The result of the operation
- Status flags : The status flags that are affected by the operation



Condition Code Register (Status Flags)

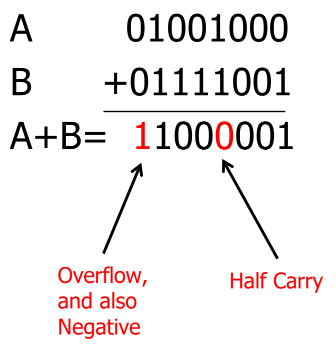
- Condition Code Register (8-bits) provides a status report on the ALU's operations
  - Carry/Borrow
  - Half carry
  - Overflow
- CCR also provides a status report after loading the Accumulator
  - Zero
  - Negative



Condition Code Register

- Arithmetic operations affect the status flags (status bits) in the CCR.

- Example:



```
A         01001000
B        +01111001
A+B=      11000001
```

Overflow, and also Negative

Half Carry

Affected Status Flags:

V=1 (overflow)
Z=0 (zero)
N=1 (negative)
H=1 (half carry)
C=0 (carry)

Registers

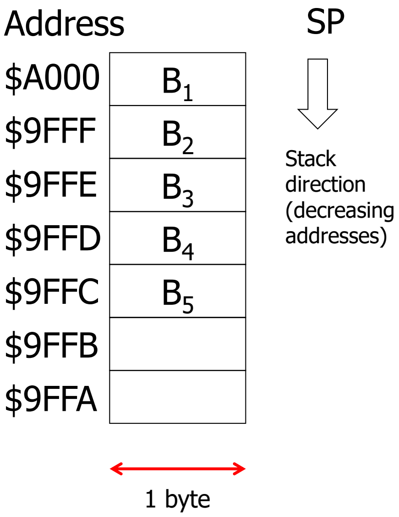
- A register is a storage location in the CPU.
- Used to hold data or a memory address during the execution of an instruction.
- Because the set of registers is small and close to the ALU, accessing data in registers is much faster than accessing data in memory outside the CPU.
- The number of registers varies from computer to computer.

Special Registers

- **Program Counter (PC) :** Holds the memory location (address) of the next instruction.
- **Instruction Register (IR) :** Holds the current instruction being executed.
- **Instruction Decoder:** It decodes the instructions and generates the control signals.
- **Index Register (IX) :** Used to keep the address (subscript) of a specified element within an array.

Stack

- **Stack Pointer** (SP) always points to the next available (empty) location in the stack.
- The stack direction is from high to low addresses.
- $ symbol is used as prefix for hexadecimal numbers.



Address	
$A000	B_1
$9FFF	B_2
$9FFE	B_3
$9FFD	B_4
$9FFC	B_5
$9FFB	
$9FFA	

System Clock

- System clock generates **pulses** to synchronize all system events.
- Each fetch-execute instruction cycle is divided into states, which are one clock pulse long.
- Most instructions require multiple steps, and so require **several clock pulses** to complete.
- Some steps (e.g. a **memory access**) take longer, and may require additional clock pulses to complete.

System Clock

- The clock speed of a CPU determines how often (frequency) a new instruction is executed.
- It is measured in
  - Mega Hertz (MHz), or
  - Giga Hertz (GHz)
- **Example:**
  1.7 GHz means that the computer executes 1.7x10^9 clock periods per second.

Units of CPU Speed

- **Hertz** (clock rate) is the unit of frequency measurement.
- It represents number of **clock cycles** per second.

Frequency Unit	Unit name	Hertz Value
1 KHz	Kilo	10^3
1 MHz	Mega	10^6
1 GHz	Giga	10^9

Time Unit	Unit name	Seconds Value
1 ms	Mili	10^{-3}
1 µs	Micro	10^{-6}
1 ns	Nano	10^{-9}

Example : CPU Clock Period

Assume the CPU clock frequency is 100 MHz.

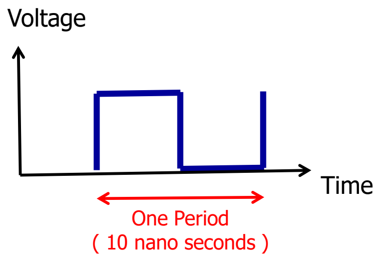
QUESTION: Find the clock period in terms of nano seconds.

$$Period(seconds) = \frac{1}{Frequency}$$

$$= \frac{1}{100*10^6} = 10^{-8}\ seconds$$

$$= 10^{-8}\ seconds\ * 10^9\ nanoseconds/second$$

$$= 10\ nanoseconds$$

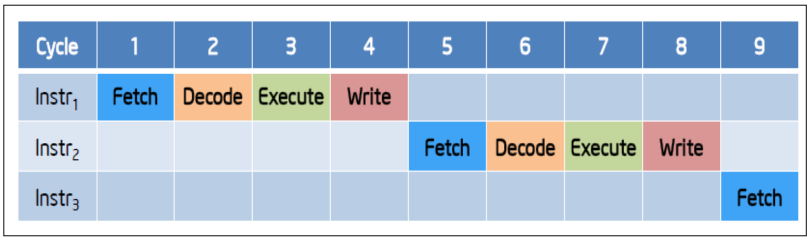


Instruction execution

- Some microprocessors can overlap the fetching, decoding and execution cycles of a number of instructions at same time.
- This is called **pipelined** (pseudo-parallel) processing.

Sequential Processing

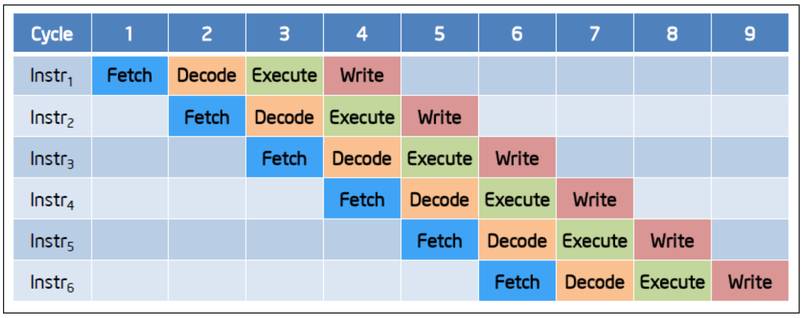
- Sequential processing works as one instruction at a time.
- Horizontal axis represents the time.
- Vertical axis represents the instructions.
- Example: 2 Instructions are processed in 8 instruction cycles (total).



Cycle	1	2	3	4	5	6	7	8	9
$Instr_1$	Fetch	Decode	Execute	Write					
$Instr_2$					Fetch	Decode	Execute	Write	
$Instr_3$									Fetch

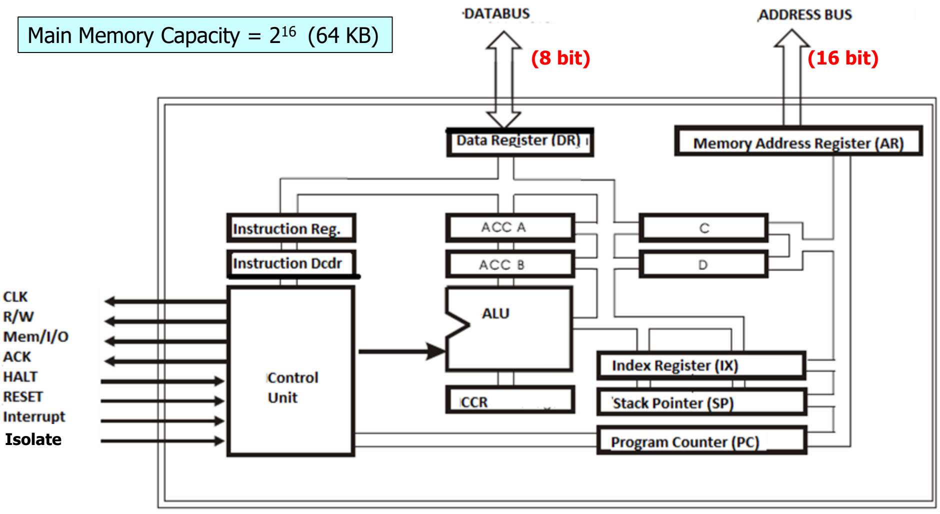
Pipelined Processing

- Pipelining improves how many tasks can be completed per unit of time.
- Example: 6 Instructions are processed in 9 instruction cycles (total).
- Each job still takes 4 cycles to complete.

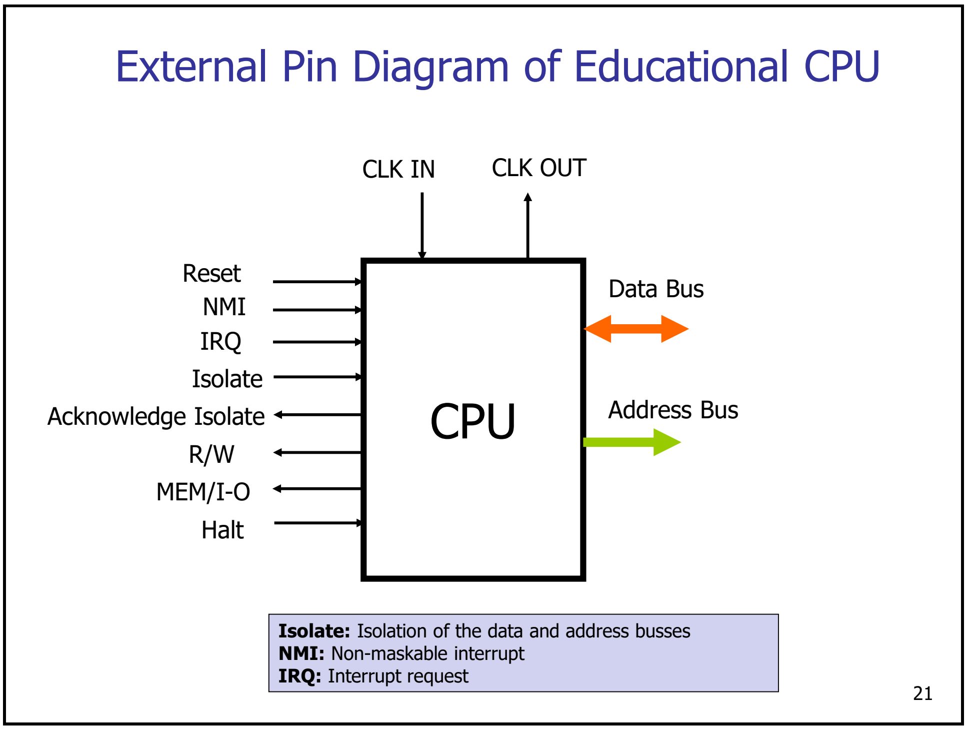


Cycle	1	2	3	4	5	6	7	8	9
Instr_1	Fetch	Decode	Execute	Write					
Instr_2		Fetch	Decode	Execute	Write				
Instr_3			Fetch	Decode	Execute	Write			
Instr_4				Fetch	Decode	Execute	Write		
Instr_5					Fetch	Decode	Execute	Write	
Instr_6						Fetch	Decode	Execute	Write

Structure of Educational CPU (Mikbil Simulator)



Main Memory Capacity = 2^{16} (64 KB)



External Pin Diagram of Educational CPU

CLK IN

CLK OUT

Reset

NMI

IRQ

Isolate

Acknowledge Isolate

R/W

MEM/I-O

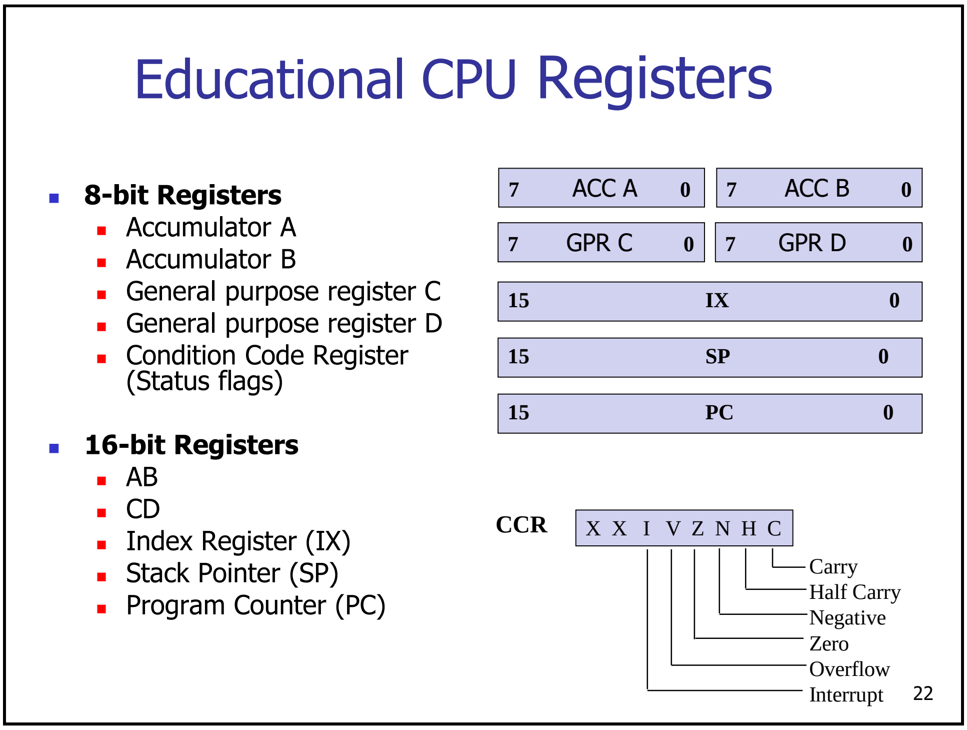
Halt

CPU

Data Bus

Address Bus

Isolate: Isolation of the data and address busses
NMI: Non-maskable interrupt
IRQ: Interrupt request



Educational CPU Registers

- **8-bit Registers**
  - Accumulator A
  - Accumulator B
  - General purpose register C
  - General purpose register D
  - Condition Code Register (Status flags)
- **16-bit Registers**
  - AB
  - CD
  - Index Register (IX)
  - Stack Pointer (SP)
  - Program Counter (PC)

7 ACC A 0 | 7 ACC B 0

7 GPR C 0 | 7 GPR D 0

15 IX 0

15 SP 0

15 PC 0

CCR X X I V Z N H C

Carry
Half Carry
Negative
Zero
Overflow
Interrupt

Register Names in Educational CPU

8 Bit Registers	
Accumulator A	A
Accumulator B	B
AUX Register C	C
AUX Register D	D
Status Register	DK (Durum Kütüğü)

16 Bit Registers	
Accumulator pair	AB
AUX register pair	CD
Index register	SK (Sıralama Kütüğü)
Stack pointer	YG (Yığın Göstergesi)
Program counter	PS (Program Sayacı)

Status Flag Bit Names
in Condition Code Register

K	T	S	N	Y	E

Durum Bayrakları (Status Flags)

K : Kesme	(Interrupt)
T : Taşma	(Overflow)
S : Sıfır	(Zero)
N : Negatif	(Negative)
Y : Yarım Elde	(Half Carry)
E : Elde	(Carry)

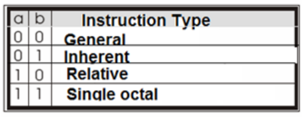
Topics

- Central Processing Unit Structure
- Instruction Format

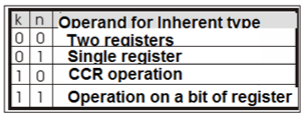
Instruction Format of EDU-CPU

An instruction format defines the layout of bits in an instruction.

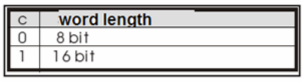
a	b	c	d	e	f	g	h	k	n	o	p	s	u	v	y	Data/address	Data/address	Address
1.Octal (byte)								2.Octal								3.Octal	4.Octal	5.Octal



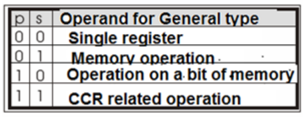
a	b	Instruction Type
0	0	General
0	1	Inherent
1	0	Relative
1	1	Single octal



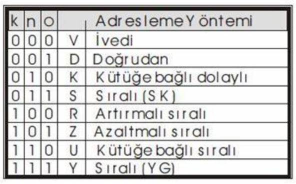
k	n	Operand for Inherent type
0	0	Two registers
0	1	Single register
1	0	CCR operation
1	1	Operation on a bit of register



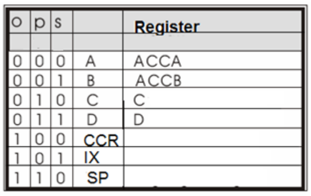
c	word length
0	8 bit
1	16 bit



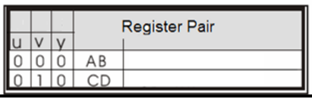
p	s	Operand for General type
0	0	Single register
0	1	Memory operation
1	0	Operation on a bit of memory
1	1	CCR related operation



k	n	o		Adresleme Yöntemi
0	0	0	V	İvedi
0	0	1	D	Doğrudan
0	1	0	K	Kütüğe bağlı dolaylı
0	1	1	S	Sıralı (SK)
1	0	0	R	Artırmalı sıralı
1	0	1	Z	Azaltmalı sıralı
1	1	0	U	Kütüğe bağlı sıralı
1	1	1	Y	Sıralı (YG)



o	p	s		Register
0	0	0	A	ACCA
0	0	1	B	ACCB
0	1	0	C	C
0	1	1	D	D
1	0	0	CCR	
1	0	1	IX	
1	1	0	SP	



u	v	y	Register Pair
0	0	0	AB
0	1	0	CD

- First two bytes are op-code.
- Others are operands.

Instruction Format

- An instruction is often divided into two parts
  - An opcode (Operation Code) that specifies the operation for that instruction
  - An address that specifies the registers and/or locations in memory to use for that operation

Op-code	Operand
1 or 2 bytes	0 to 3 bytes

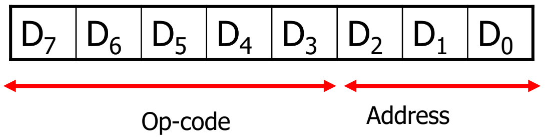
Total instruction length :
Min 1 byte ,
Max 5 bytes

Single Word, 1-Address Instruction Format

- **8-bit words**

Op-code is 5 bits ---> 2^5 = 32 possible op-codes.

Address is 3 bits ---> 2^3 = 8 possible addresses.
(Enough for register operations.)



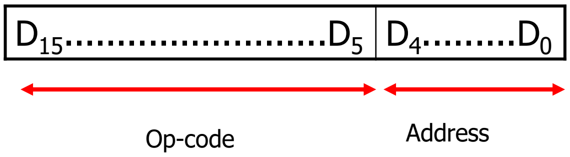
Single Word, 1-Address Instruction Format

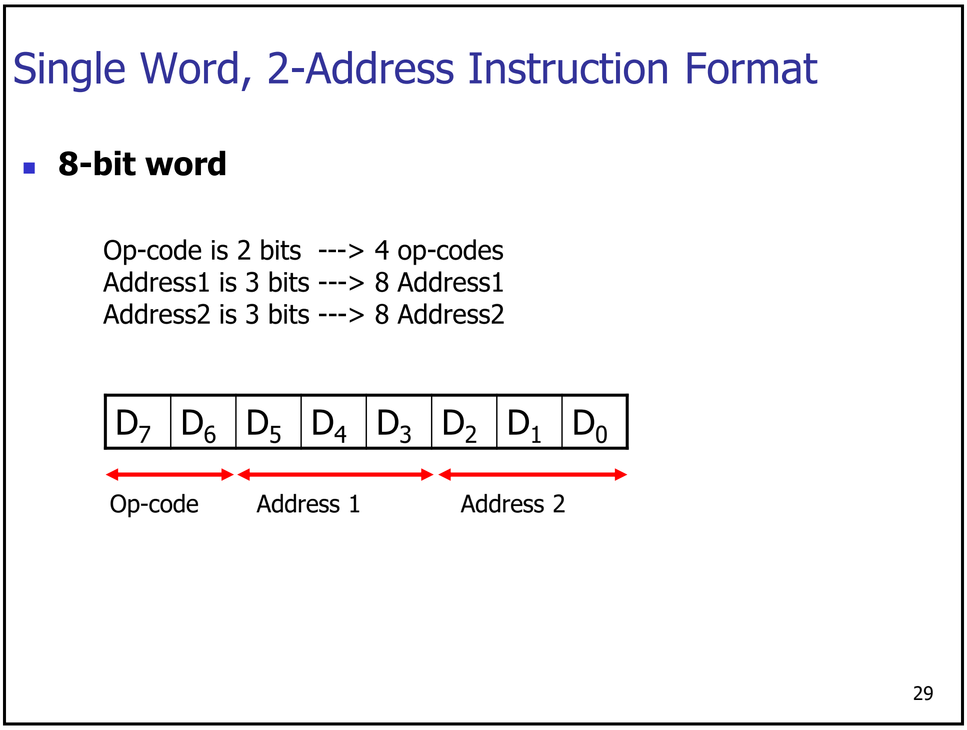
- **16-bit words**

Op-code is 11 bits ---> 2048 possible op-codes,

Address is 5 bits ---> 32 possible adresses.

More operation code and addressing possibilities.





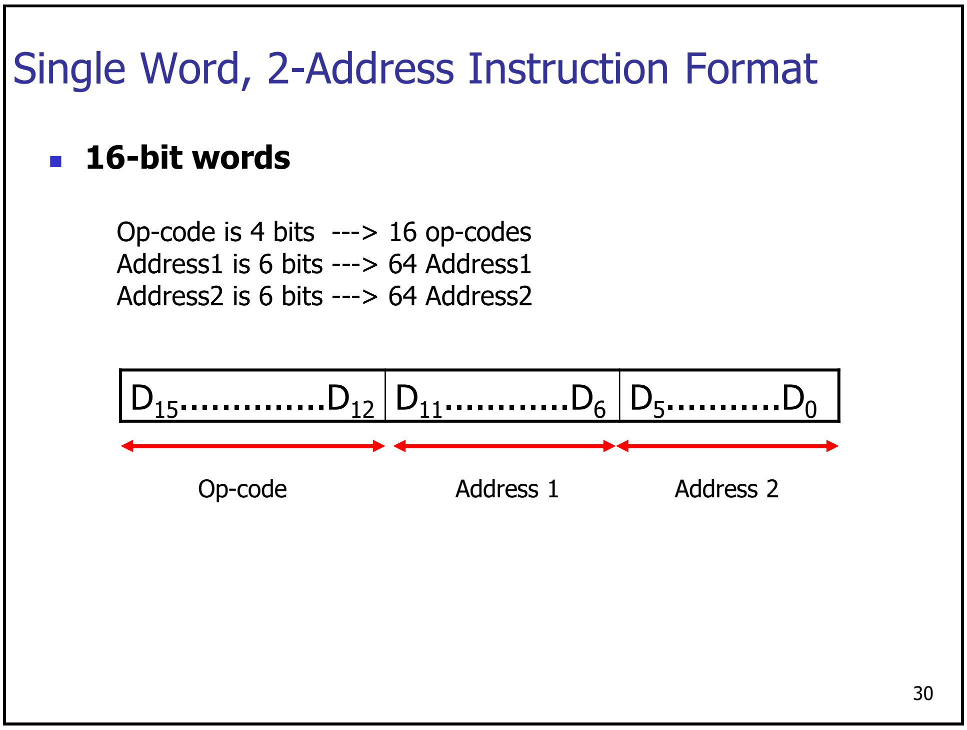
Single Word, 2-Address Instruction Format

- **8-bit word**

Op-code is 2 bits ---> 4 op-codes
Address1 is 3 bits ---> 8 Address1
Address2 is 3 bits ---> 8 Address2

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Op-code   Address 1   Address 2



Single Word, 2-Address Instruction Format

- **16-bit words**

Op-code is 4 bits ---> 16 op-codes
Address1 is 6 bits ---> 64 Address1
Address2 is 6 bits ---> 64 Address2

D_{15}..............D_{12}	D_{11}............D_6	D_5...........D_0

Op-code   Address 1   Address 2

Multiple Words, 1-Address Instruction Format

- 1-Address instruction in multiple words (**24-bit words**)

Op-code is 8 bits ---> 256 op-codes
Address is 16 bits ---> 64K Addresses

1. Octal	2. Octal	3. Octal
Operation Code (Op-code)	Upper half of the address (High byte)	Lower half of the address (Low byte)

Examples of N-Byte Instructions

- The followings are some examples of Assembly instructions (EDU-CPU) that have different byte lengths.
- Total byte length of an instruction is the sum of opcode part and operands part.

Example	Memory Adress (Hex)	Machine Codes (Hexadecimal)					Assembly Instruction	Explanation
		Opcode Bytes		Operand Bytes				
1-Byte Instruction	0000	C3					INT	Interrupt (stop)
2-Byte Instruction	0001	4B	40				CLR A	Clear A accumulator
3-Byte Instruction	0003	00	00	58			LDA A, $58	Load A with immediate data
4-Byte Instruction	0006	00	20	30	00		LDA A, <$3000>	Load A from address
5-Byte Instruction	000A	01	08	95	20	00	STA $95, <$2000>	Store immediate data to address

Instruction Sets

- Depending on the architecture, the instruction set is organized as :
- **CISC (Complex Instruction Set Computer):**
  - Contains large number of instructions
  - More complex on hardware
  - Mostly used in **Von-Neumann** architecture processors
  - **Examples:**
    - Motorola (MC68K)
    - Intel 80xx
- **RISC (Reduced Instruction Set Computer):**
  - Contains fewer but effective instructions
  - More complex on software
  - Mostly used in **Harvard** architecture processors
  - **Examples:**
    - ARM Cortex
    - **Educational CPU**
      (Its architecture is Von-Neumann.)

Instruction Set Differences

- Consider Z = X + Y instruction in a high level language.
- **CISC architecture**
  It might be translated into **one** Assembly instruction (pseudocode).

```
add mem(X), mem(Y), mem(Z)
;Add memory X and Y, result is in memory Z
```

- **RISC architecture**
  Four Assembly instructions (pseudocodes).

```
load R1, X          ;Load Register1 from memory X
load R2, Y          ;Load Register2 from memory Y
add R3, R1, R2      ;Add Register1 and Register2,
                    ;result is in Register3
store Z, R3         ;Store Register3 to memory Z
```

Instruction Categories

- Transfer instructions: Data transfers among registers, or registers and main memory
  - Load, Store, Exchange
- Arithmetic, logic, and shift instructions:
  - Add, Complement, Increment, Rotate, Shift, AND, OR, Clear, Set,etc.
- Program control instructions, and instructions to check status conditions:
  - Compare, Branch, Jump
- Input/Output Instructions (for peripheral devices):
  - Input data, Output data

Assembly Language and Machine Language Example

Assembly language instruction template syntax (MİKBİL)

```
{Label}     Operation,     Operand      ; {Explanation}
```

Assembly language codes

```
START   LDA  A, <$0080>     ; Load A accumulator register
                            ; with number from memory address $0080

        ADD  A, <$0081>     ; Add to A the number from address $0081

        STA  A, <$0082>     ; Store A to address $0082
        INT                 ; Interrupt (Stop)
```

Machine language codes (hexadecimal)

Address	Content
0000	00 20 00 80
0004	03 20 00 81
0008	01 20 00 82
000C	C3