به نام خدا

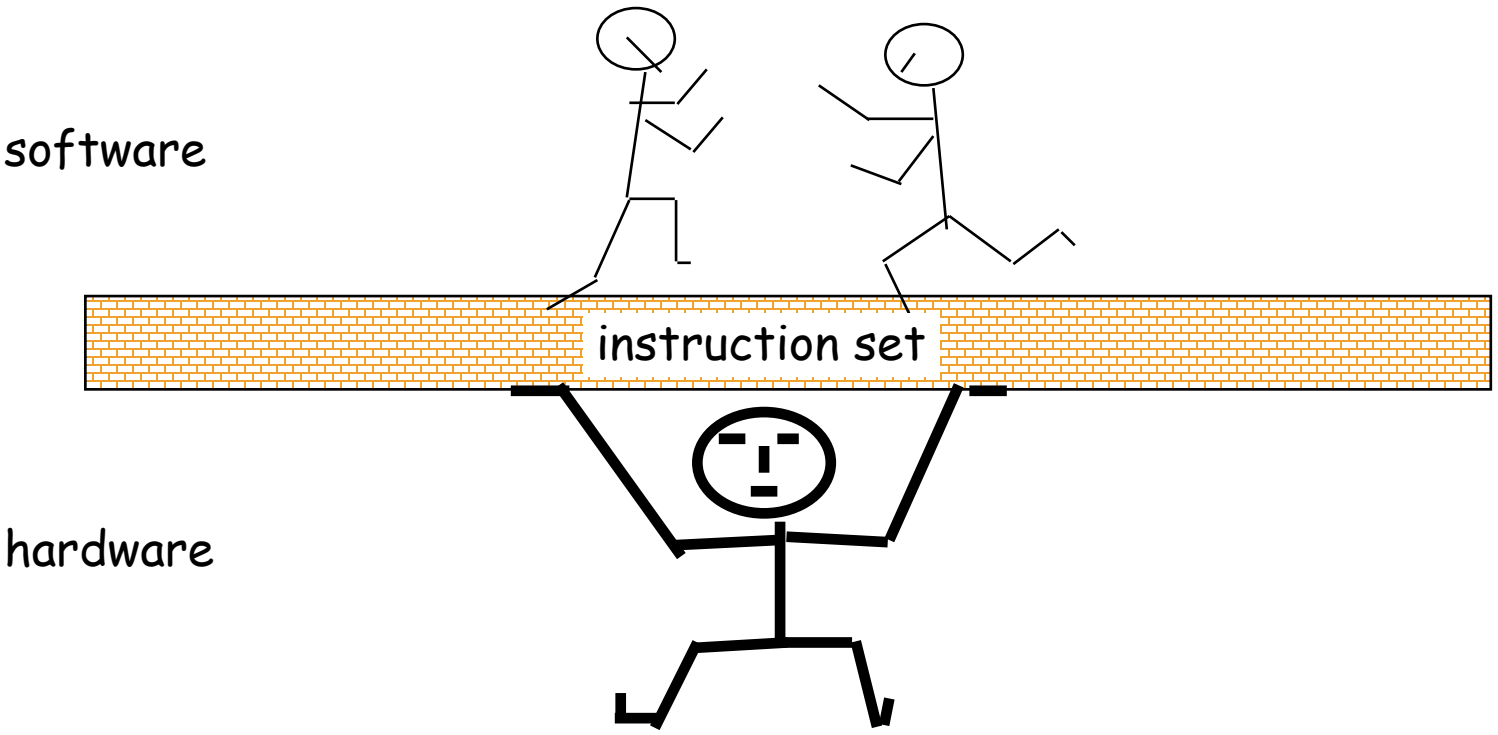
کامپیوتر پایه

بررسی ISA

Dr. Aref Karimiafshar
A.karimiafshar@iut.ac.ir

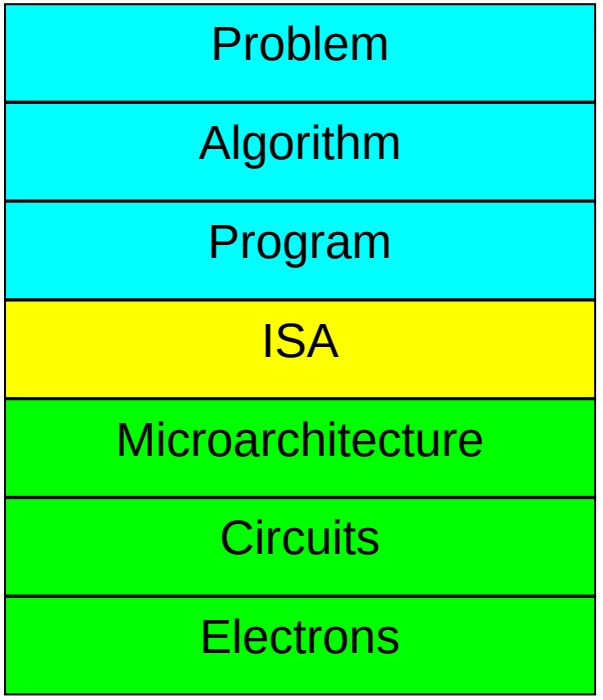
The Instruction Set: a Critical Interface

The actual programmer visible instruction set



ISA vs. Microarchitecture

- ISA
  - Agreed upon interface between software and hardware
    - SW/compiler assumes, HW promises
  - What the software writer needs to know to write and debug system/user programs
- Microarchitecture
  - Specific implementation of an ISA
  - Not visible to the software



ISA

- Instructions
  - Opcodes, Addressing Modes, Data Types
  - Instruction Types and Formats
  - Registers, Condition Codes
- Memory
  - Address space, Addressability, Alignment
  - Virtual memory management
- I/O: memory-mapped vs. instr.

Intel® 64 and IA-32 Architectures
Software Developer's Manual

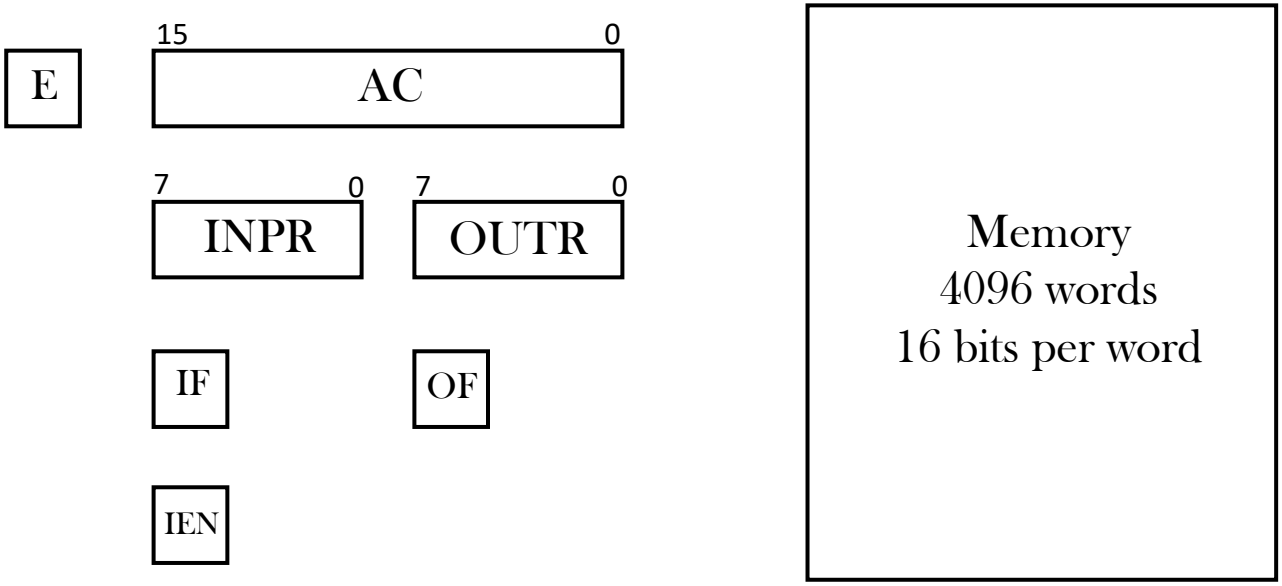
Volume 1:
Basic Architecture

Many Different ISAs Over Decades

- x86
- PDP-x: Programmed Data Processor (PDP-11)
- VAX
- IBM 360
- CDC 6600
- SIMD ISAs: CRAY-1, Connection Machine
- VLIW ISAs: Multiflow, Cydrome, IA-64 (EPIC)
- PowerPC, POWER
- RISC ISAs: Alpha, MIPS, SPARC, ARM

- What are the fundamental differences?
  - E.g., how instructions are specified and what they do
  - E.g., how complex are the instructions

Basic Computer



Basic Computer

- PDP-8 is a 12-bit minicomputer
  - produced by Digital Equipment Corporation (DEC)
  - first commercially successful minicomputer

Basic Computer Instructions

- **Memory-reference Instruction**
  - Opcode = 000 ~ 110

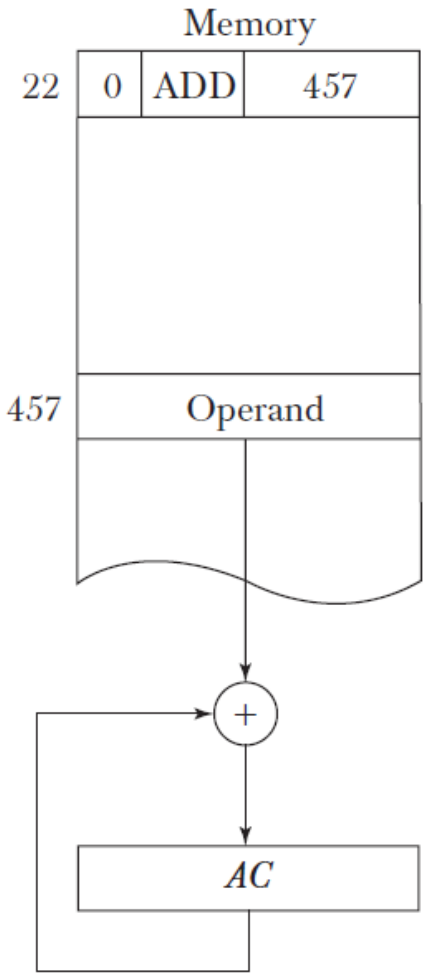
- **Register-reference Instruction**

- **Input-Output Instruction**

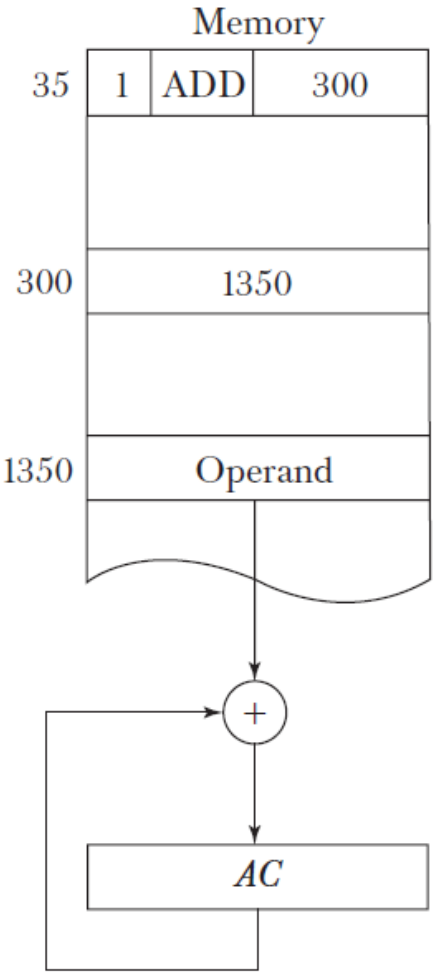
Memory-reference Instructions

Symbol	Code		Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	ADD memory word to AC
LDA	2xxx	Axxx	Load memory word to AC
STA	3xxx	Bxxx	Store Content of AC in memory
BUN	4xxx	Cxxx	Branch Unconditionally
BSA	5xxx	Dxxx	Branch and Save Return Address
ISZ	6xxx	Exxx	Increment and Skip if Zero

Addressing Modes



Direct address



Indirect address

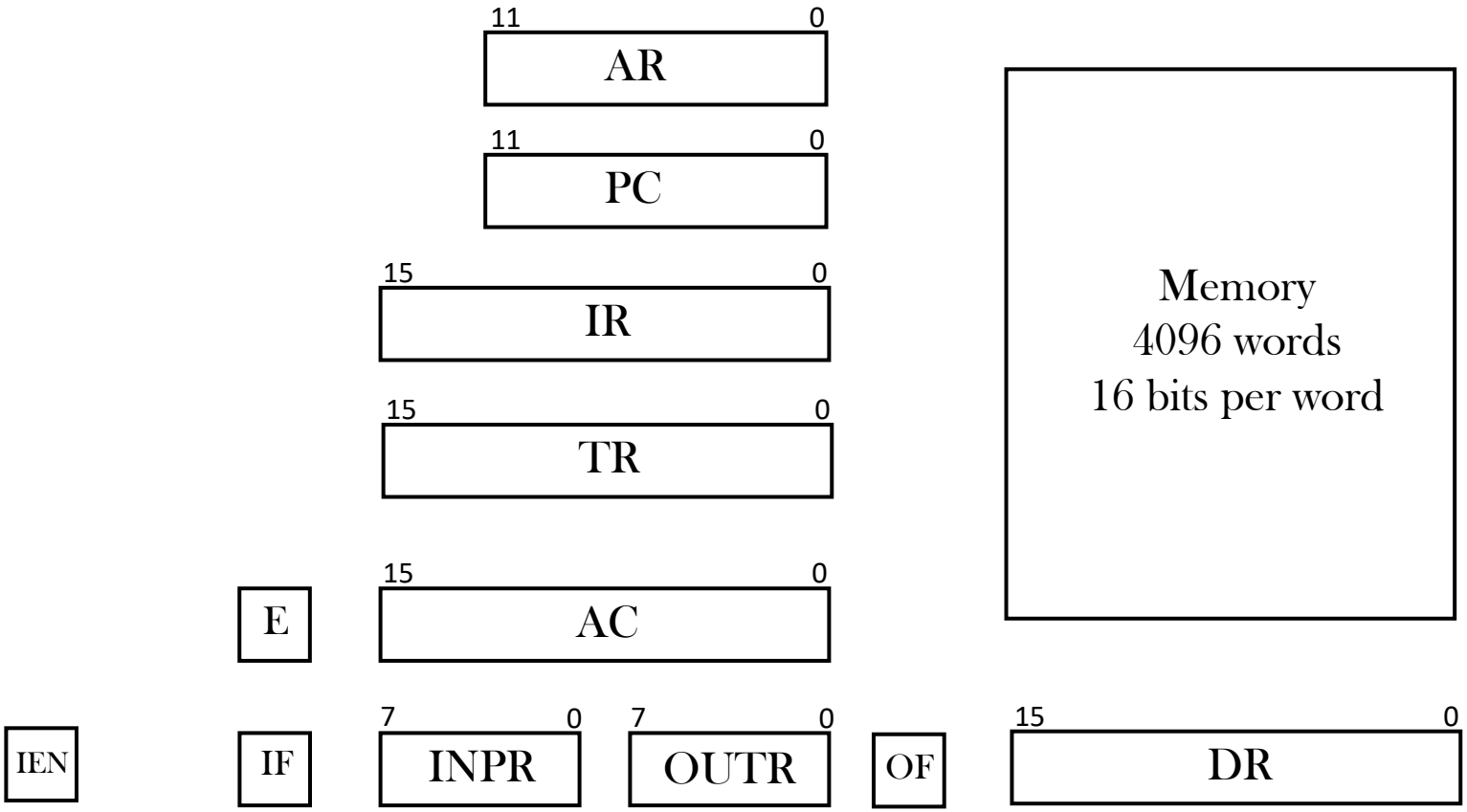
Register-reference Instructions

Symbol	Code	Description
CLA	7800	Clear AC
CLE	7400	Clear E
CMA	7200	Complement AC
CME	7100	Complement E
CIR	7080	Circulate Right AC and E
CIL	7040	Circulate Left AC and E
INC	7020	Increment AC
SPA	7010	Skip next instruction if AC positive
SNA	7008	Skip next instruction if AC negative
SZA	7004	Skip next instruction if AC zero
SZE	7002	Skip next instruction if E is zero
HLT	7001	Halt Computer

Input-Output Instructions

Symbol	Code	Description
INP	F800	Input Character to AC
OUT	F400	Output Character from AC
SKI	F200	Skip on input flag
SKO	F100	Skip on output flag
ION	F080	Interrupt ON
IOF	F040	Interrupt OFF

Basic Computer



Machine Language

- Program
  - A list of Instructions or Statement for directing the computer to perform a required data processing
- Machine Language Program
  - Binary Code
  - But for convenience, Hexadecimal Code is preferred
  - Symbolic Code or Assembly Code is easy to understand

Assembly Language for Basic Computer

- Each line of Assembly Language program is arranged in three Fields
  - Label field: Empty or Symbolic Address
  - Instruction field: Machine instruction or Pseudo instruction
  - Comment field: Empty or Comments
- Ex:

Label	Instruction	Comment
	LDA X	
	LDA X	/Load X to AC
ONE,	LDA X	/Load X to AC

Assembly Language (Cont.)

- Symbolic Address
  - Not more than Three Alphanumeric characters
  - First Character should be Letter
  - Terminated by Comma
- Comments
  - Start with Slash (/)
  - Explaining the program, Easy to understand
  - Not Converted to Binary Code

Assembly Language (Cont.)

- Instruction Field
  - MRI (Memory Reference Instruction)
    - Symbol for MRI Symbolic Address Indirect

      ADD OPR

      ADD OPR I

    MUST occurs at Label Field 
  - Non-MRI: Register reference or I/O Instruction
    - Symbol for MRI NO Symbolic Address

      CLA
  - Pseudo instruction with or without Operand

Assembly Language (Cont.)

- Pseudo instruction
  - Not Machine instruction but instruction for assembler control

Symbol	Information for the assembler
ORG N	Hexadecimal N is the memory location for the Instruction or Operand listed in the following line
END	Denotes the End of symbolic Program
DEC N	Signed decimal number N to be converted to Binary
HEX N	Hexadecimal number N to be converted to Binary

Assembly Language: Example

Type	Label	Instruction	Comments
PseudoIns.		ORG 100	/Origin of program is location 100H
MRI		LDA SUB	/Load SUB to AC
Non-MRI		CMA	/Complement AC
Non_MRI		INC	/Increment AC
MRI		ADD MIN	/Add MIN to AC
MRI		STA DIF	/Store AC to DIF
PseudoIns.		HLT	/Halt Computer
PseudoIns.	MIN,	DEC 83	/MIN value
PseudoIns.	SUB,	DEC -23	/SUB value
PseudoIns.	DIF,	HEX 0	/DIF will saved here
PseudoIns.		END	/End of Program

Translation to Binary

- Assembler translate Assembly Language (Source Program) to Binary Code (Object Program)
  - For more information,
    - 1st Pass: Generate Address Symbol Table
    - 2nd Pass: Machine instructions are translated by means of table lookup procedure
      - Pseudo instruction Table
      - MRI Table
      - Non-MRI Table
      - Address Symbol Table
    - Error Diagnostics
      - Syntax error, Not defined symbolic address, ...

Translation to Binary (Cont.)

Assembly		Hexadecimal		Microoperation
ORG 100				PC ← 100
LDA	SUB	100	2107	AR ← PC, PC ←PC+1,…
CMA		101	7200	
INC		102	7020	
ADD	MIN	103	1106	
STA	DIF	104	3108	
HLT		105	7001	
MIN,	DEC 83	106	0053	
SUB,	DEC -23	107	FFE9	
DIF,	HEX 0	108	0000	
END				

Program Loops

- Program Loop
  - A sequence of instructions that are executed many times, each time with a different set of data
  - Example (SUM = A(1)+…+A(100))

```
int A[100];
int i,  sum=0;
for(i=0; i<100; i++)
{
     SUM = SUM + A[i];
 }
```

Program Loops (cont.)

```
        ORG 100
        LDA ADS
        STA PTR
        LDA NBR
        STA CTR
        CLA
LOP,    ADD PTR I
        ISZ PTR
        ISZ CTR
        BUN LOP
        STA SUM
        HLT
ADS,    HEX 150
PTR,    HEX 0
NBR,    DEC -100
CTR,    HEX 0
SUM     HEX 0
```

100	LDA 10B
101	STA 10C
102	LDA 10D
103	STA 10E
104	CLA
105	ADD 10C I
106	ISZ 10C
107	ISZ 10E
108	BUN 105
109	STA 10F
10A	HLT
10B	HEX 150
10C	HEX 0
10D	DEC -100
10E	HEX 0
10F	HEX 0

Can you trace this program ?
Esp. AC, PTR and CTR ?

```
ORG 150
DEC 75
.
.
.
DEC 23
END
```

150	DEC 75
…	…
1B4	DEC 23

Program Loops (cont.)

	ORG 100				
	LDA ADS	AC←150H			
	STA PTR	PTR←150H			
	LDA NBR	AC← -100D			
	STA CTR	CTR← -100D			
	CLA	AC←0H			
LOP,	ADD PTR I	AC←0+(150H)=0+75D	AC←75D+(151H)	…	AC←AC+23D
	ISZ PTR	PTR←150H+1	PTR←151H+1	…	PTR←1B4H+1
	ISZ CTR	CTR← -100D+1	CTR← -99D+1	…	CTR← -1D+1
	BUN LOP	PC←LOP	PC ← LOP		SKIP
	STA SUM				SUM←AC
	HLT				HALT
ADS,	HEX 150	ORG 150			
PTR,	HEX 0	DEC 75			
NBR,	DEC -100	.			
CTR,	HEX 0	.			
SUM	HEX 0	.			
		DEC 23			
		END			

Programming Arithmetic and Logic Operations

- Implementation of Arithmetic and Logic Op
  - By Hardware
    - Complex ALU, but Fast
    - One Machine instruction for operation
  - By Software
    - Simple ALU, but Slow
    - A set of instructions for operation
    - Our Basic Computer has Only one arithmetic instruction (ADD), Others are programmed using basic instruction

Multiplication Program

- Specification of Program
  - Multiplication of 8-bit positive numbers
    - 8-bit x 8-bit = 16-bit Result
- Ex: 0000 1111B x 0000 1011B
    - 00001111 x (00000001 + 00000010 + 00001000)

    = 00001111 + 00001111 x 2 + 00001111 x 8

    = 00001111 + 00001111 << 1 + 00001111 << 3

    = 00001111 + 00011110 + 01111000

    = 10100101
    - Multiplication can be implemented with SHIFT and ADD instructions

Multiplication Program (cont.)

	ORG	100			
LOP,	CLE		E←0	E←0	E←0
	LDA	Y	AC←000B	AC←0005	AC←0002
	CIR		E←1, AC←0005	E←1, AC←0002	E←0, AC←0001
	STA	Y	Y←0005	Y←0002	Y←0002
	SZE				
	BUN	ONE	PC←ONE	PC←ONE	
	BUN	ZRO			PC←ZRO
ONE,	LDA	X	AC←000F	AC←001E	
	ADD	P	AC←000F+0	AC←001E+000F	
	STA	P	P←AC	P←AC	
	CLE		E←0	E←0	
ZRO,	LDA	X	AC←000F	AC←001E	AC←003C
	CIL		E←0, AC←001E	E←0, AC←003C	E←0, AC←0078
	STA	X	X←AC	X←AC	X←AC
	ISZ	CTR	CTR←1-8	CTR←1-7	CTR←1-6
	BUN	LOP	PC←LOP	PC←LOP	PC←LOP
	HLT				
CTR,	DEC	-8	-7	-6	-5
X,	HEX	000F	001E	003C	0078
Y,	HEX	000B	0005	0002	0001
P,	HEX	0	000F	002D	002D
	END				

Double Precision Addition

- Double Precision
  - A number stored in Two memory word
  - Used for great accuracy
  - Ex: 00F0FF0F + 00FF00FF = 01F0000E

```
        ORG     100
        LDA     AL      AC←FF0F
        ADD     BL      E←1, AC←000E (1000E=FF0F+00FF)
        STA     CL      CL←000E
        CLA             AC←0
        CIL             E←0, AC←0001
        ADD     AH      AC←0001+00F0
        ADD     BH      AC←00F1+00FF
        STA     CH      CH←01F0
        HLT
AL,     HEX     FF0F
AH,     HEX     00F0
BL,     HEX     00FF
BH,     HEX     00FF
CL,     HEX     0       000E
CH,     HEX     0       01F0
        END
```

Logic Operations

- Basic Computer has AND, CMA, CLA
- Implementation of OR
  - DeMorgan's theorem: A # B = (A' & B')'

```
LDA     A           /AC←A
CMA                 /AC←A'
STA     TMP         /TMP←A'
LDA     B           /AC←B
CMA                 /AC←B'
AND     TMP         /AC←B'&A'
CMA                 /AC←(B'&A')'
```

Shift Operations

- Basic Computer has CIR, CIL

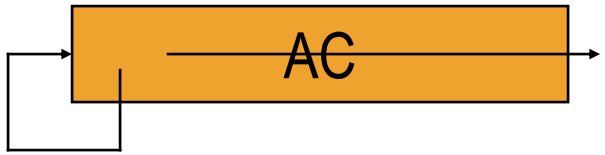
- Implementation of Logical Shift

shr: CLE
CIR

shl: CLE
CIL

Shift Operations

- Implementation of Arithmetic Right Shift




```
ashr:   CLE   /E←0
        SPA   /if AC>0 (MSB of AC is 0), Skip
        CME   /else E←1
        CIR   /Circulate E and AC
```

Subroutines

- Subroutine
  - A set of common instructions that can be used in program many times
  - Basic Computer has BSA instruction
    - Branch and Save Return Address
- MACRO
  - Assembler Utility for not common but Similar instructions

Subroutine: Example

```
        ORG     100
100     LDA     X       AC←1234
101     BSA     SH4     SH4←102(=PC), PC←10A
102     STA     X       X←AC
103     LDA     Y       AC←4321
104     BSA     SH4     SH4←105(=PC), PC←10A
105     STA     Y       Y←AC
106     HLT
107 X,  HEX     1234
108 Y,  HEX     4321
```

```
109 SH4, HEX    0       102             105
10A     CIL             AC←cil(AC)      AC←cil(AC)
10B     CIL             AC←cil(AC)      AC←cil(AC)
10C     CIL             AC←cil(AC)      AC←cil(AC)
10D     CIL             AC←cil(AC)      AC←cil(AC)
10E     AND     MSK     AC←AC & FFF0    AC←AC & FFF0
10F     BUN     SH4 I   PC←(109)=102    PC←(109)=105
110 MSK,HEX     FFF0
        END
```

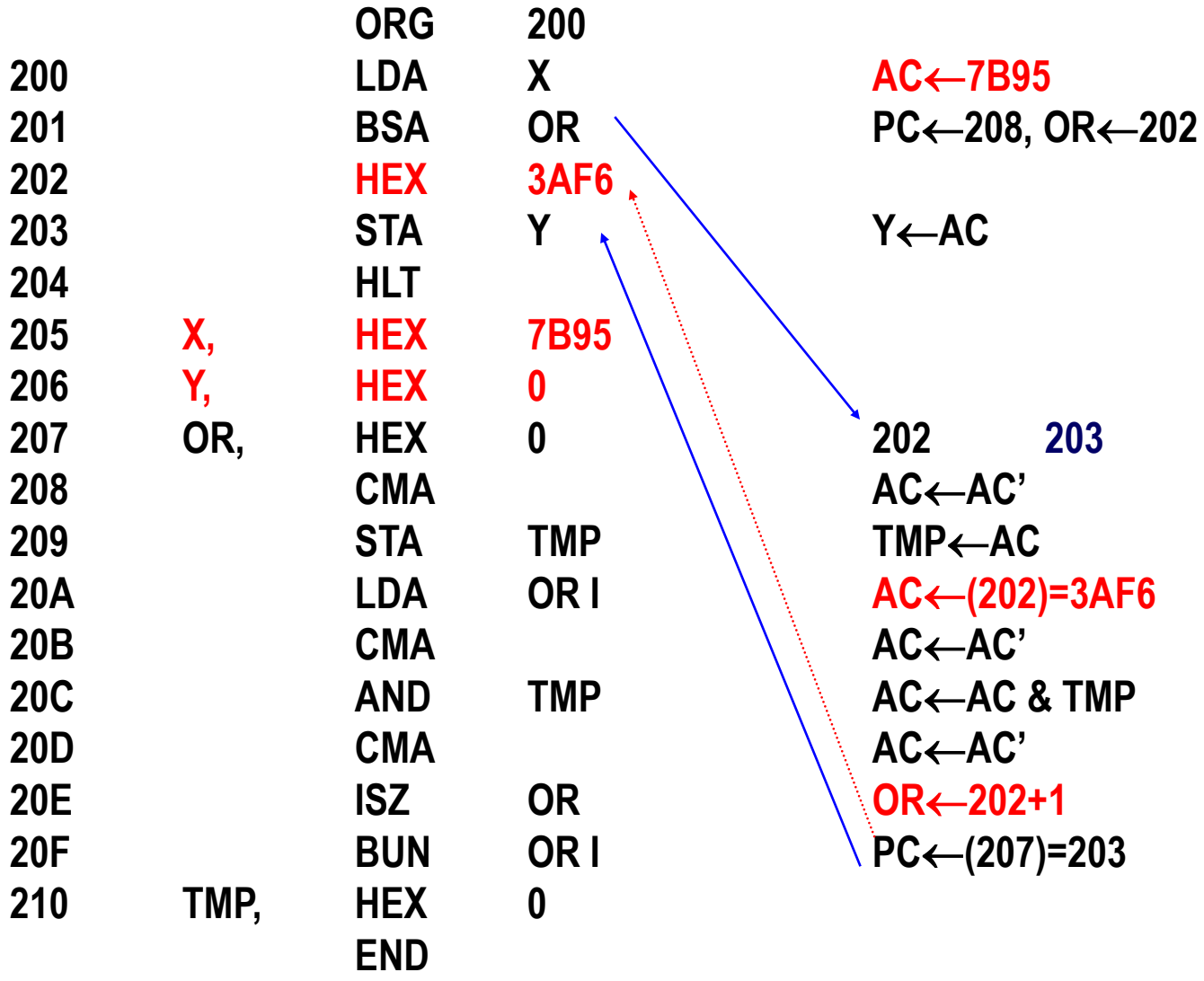
Subroutine Linkage

- Subroutine Linkage
  - Procedure for branching subroutine and returning to main program
- Basic Computer (Only ONE register)
  - BSA SUB --> call
  - BUN SUB I --> return
- Generally
  - Stack and Stack Pointer is used
    - Stack saves Return address
    - Stack Pointer is Index Register pointing the location of Stack

Subroutine Parameters and Data Linkage

- Subroutine needs Input(Output) Parameters from(to) Main routine
  - Using Registers
    - Basic Computer has AC only
      - One Input - One Output
    - The more register, The Easier
  - Using Memory
    - Basic Computer: Predefined Location
    - Generally Stack is used for parameter passing

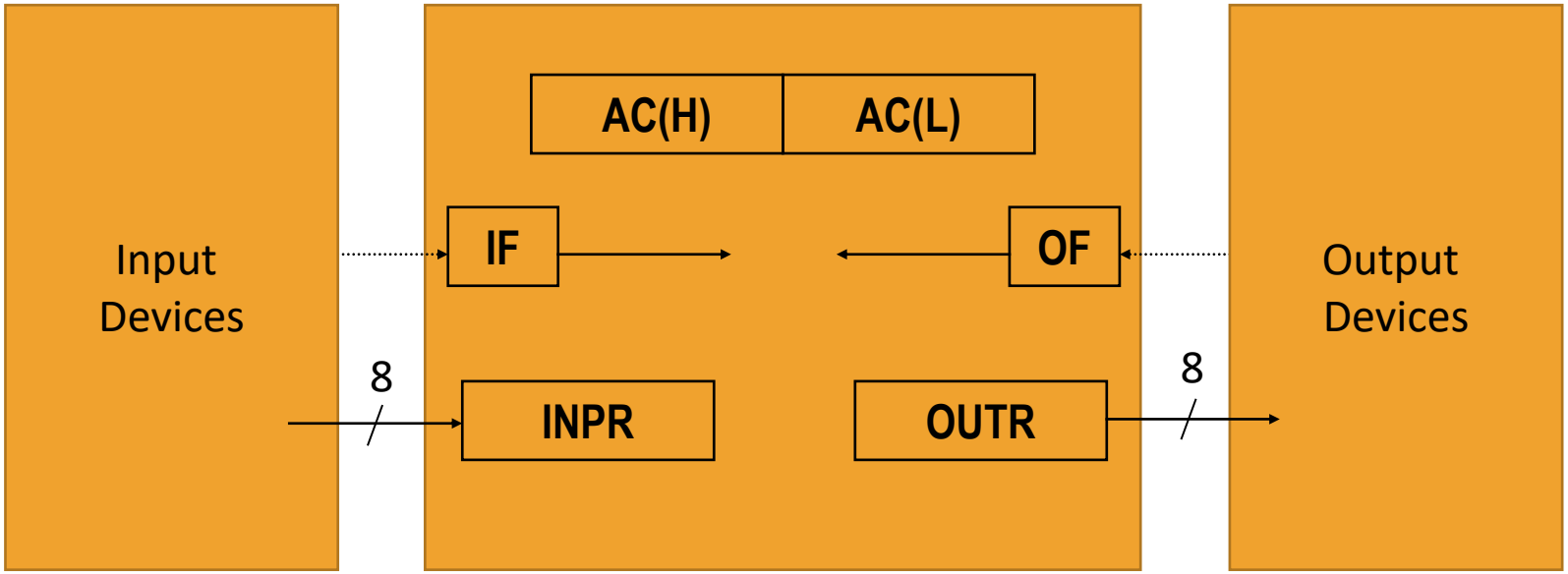
Ex: Parameter Passing



Ex: Move a Block of Data

100	BSA	MVE	MVE←101, PC←MVE+1			
101	HEX	100				
102	HEX	200				
103	DEC	-16				
104	HLT					
MVE,	HEX	0	101	102	103	104
	LDA	MVE I	AC←(101)=100			
	STA	PT1	PT1←AC			
	ISZ	MVE	MVE←101+1			
	LDA	MVE I	AC←(102)=200			
	STA	PT2	PT2←AC			
	ISZ	MVE	MVE←102+1			
	LDA	MVE I	AC←(103)=-16			
	STA	CTR	CTR←AC			
	ISZ	MVE	MVE←103+1			
LOP,	LDA	PT1 I	AC←(100)		…	AC←(10F)
	STA	PT2 I	(200)←AC		…	(20F)←AC
	ISZ	PT1	PT1←100+1		…	PT1←110
	ISZ	PT2	PT2←200+1		…	PT2←210
	ISZ	CTR	CTR←1-16		…	CTR←0
	BUN	LOP	PC←LOP			
	BUN	MVE I				PC←(MVE)=104
PT1,	HEX	0	100	101	… 10F	110
PT2,	HEX	0	200	201	… 20F	210
CTR,	HEX	0	-16D	-15D	… -1	0

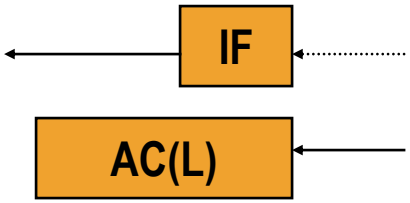
Input-Output Programming



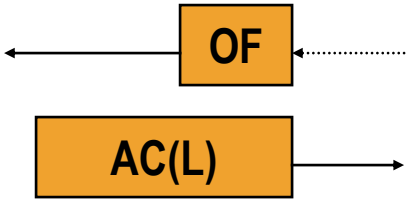
Input-Output Programming

- Programmed I/O
  - 8 bit Input-Output

```
CIF,    SKI          /Check Input Flag
        BUN    CIF   /Flag=0, No Input
        INP          /Flag=1, Read Character
        OUT          /Echo Character
        STA    CHR   /Store to CHR
        HLT
CHR,    HEX    0
```




```
        LDA    CHR   /AC←'W'
COF,    SKO          /Check Output Flag
        BUN    COF   /Flag=0, Not Ready
        OUT          /Flag=1, Output Character
        HLT
CHR,    HEX    0057  /'W'
```



Character Manipulation

- Program to Store Input Characters in a Buffer
  - Buffer Address is 500H

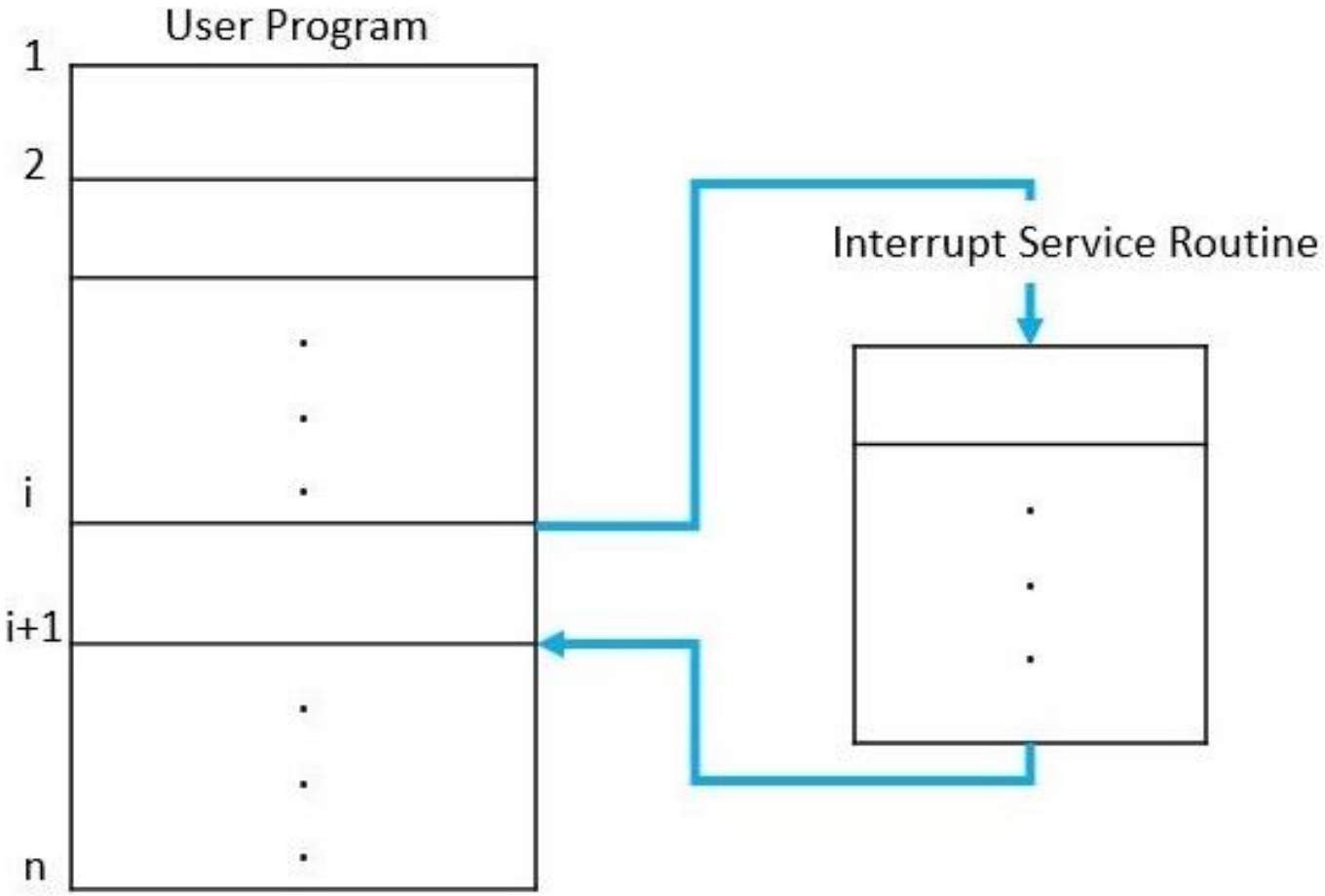
```
        LDA     ADS     /AC←500
        STA     PTR     /PTR←500
LOP,    BSA     IN2     /AC←2 character
        STA     PTR I   /(PTR)←AC
        ISZ     PTR     /PTR←PTR+1
        BUN     LOP     /PC←LOP
        HLT
ADS,    HEX     500
PTR,    HEX     0
```

```
IN2,    HEX     0
FST,    SKI
        BUN     FST
        INP
        OUT
        BSA     SH4
        BSA     SH4
SCD,    SKI
        BUN     SCD
        INP
        OUT
        BUN     IN2 I
```

Program Interrupt

- Programmed IO
  - Most of CPU time is spent at checking IF or OF
    - Loss of CPU time
    - Easy to Implement
- Interrupt
  - CPU request I/O to device
  - CPU goes to do Other jobs
  - Device Interrupt CPU when I/O operation is Done
  - CPU performs Interrupt Service Routine

Interrupt



Interrupt Service Routine

- Interrupt Occurs
  - for Interrupt cycle
    - IEN←0, (0)←PC, PC←1
- Save Contents of processor registers
  - Basic Computer has AC and E registers
- Check which Flag is Set
  - IF(FGI) and OF(FGO) Flag
  - Priority of Interrupt: Which is serviced first?
- Service the device whose flag is set
- Restore Contents of processor registers
- Turn the Interrupt facility On
  - IEN←1, for Next interrupt
- Return to the running program

Ex: ISR

```
100     CLA
101     ION
102     LDA     X
103     ADD     Y
104     STA     Z
…
```

Interrupt Occurs

```
0       ZRO,    HEX     0
1               BUN     SRV
…
200     SRV,    STA     SAC
                CIR
                STA     SE
                SKI
                BUN     NXT
                INP
                OUT
                STA     PT1 I
                ISZ     PT1
        NXT,    SKO
                BUN     EXT
                LDA     PT2 I
                OUT
                ISZ     PT2
        EXT,    LDA     SE
                CIL
                LDA     SAC
                ION
                BUN     ZRO I
        SAC,
        SE,
        PT1,
        PT2,
```

پایان

موفق و پیروز باشید