8085 Microprocessors: Organization and ISA

Virendra Singh
Associate Professor
Computer Architecture and Dependable Systems Lab
Department of Electrical Engineering
Indian Institute of Technology Bombay
http://www.ee.iitb.ac.in/~viren/
E-mail: viren@ee.iitb.ac.in

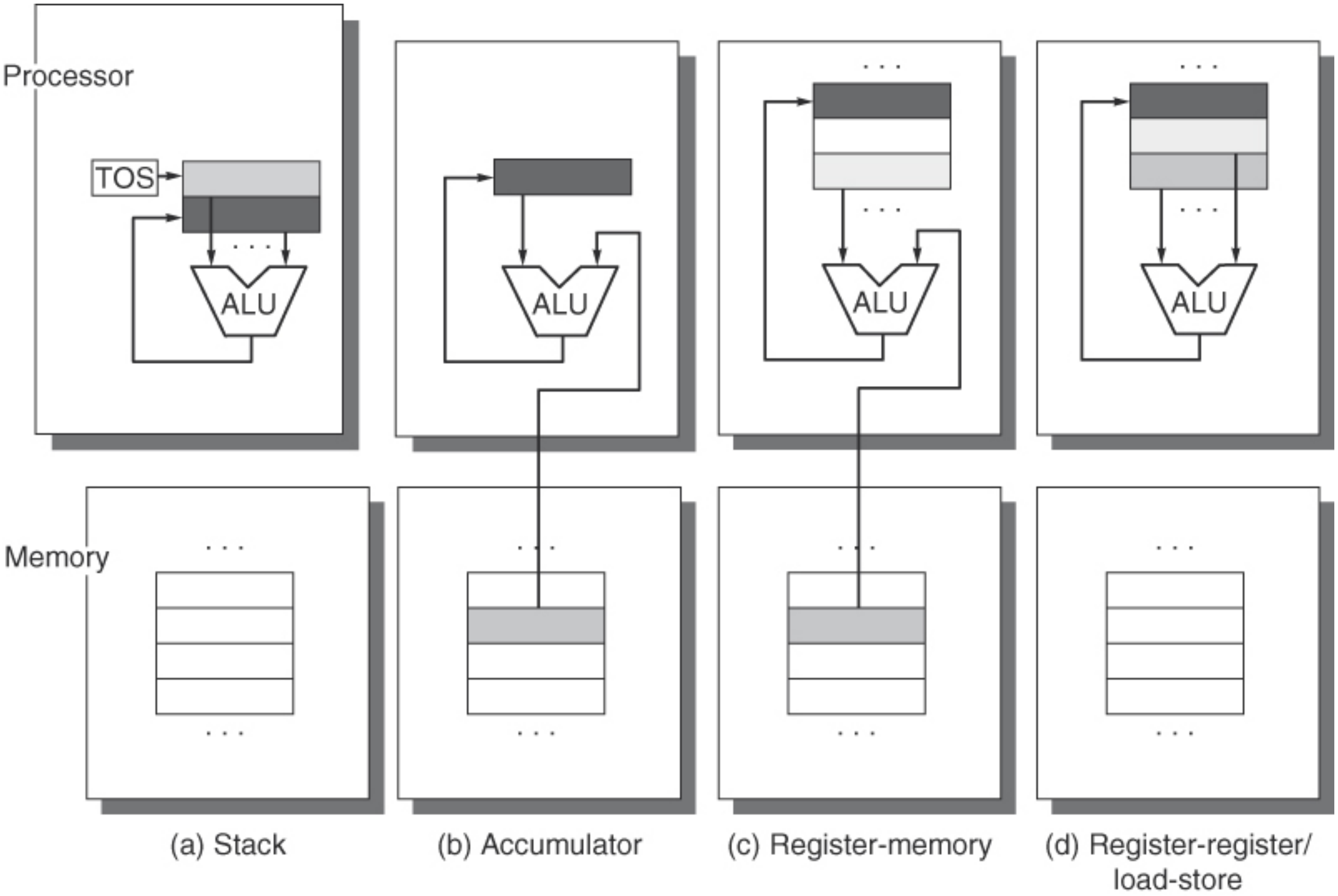
EE-309: Microprocessors

Lecture 3 (24 July 2014)

CADSL

INSTRUCTION SET ARCHITECTURE

Basic Machine Organizations



Source: CA: A quantitative approach

Register-Memory Architectures

- Instruction set:

```
add R1,  A            sub R1, A     mul R1, B
load R1, A            store R1, A
```

- Example: A*B - (A+C*B)

```
load R1, A
mul R1, B             /*    A*B             */
store R1, D
load R2, C
mul R2, B             /*    C*B             */
add R2, A             /*    A + CB               */
sub R2, D             /*    AB - (A + C*B)       */
```

Register-Memory Architectures

- Instruction set:

```
add R1,  A              sub R1, A    mul R1, B
load R1, A              store R1, A
```

- Example: A*B - (A+C*B)

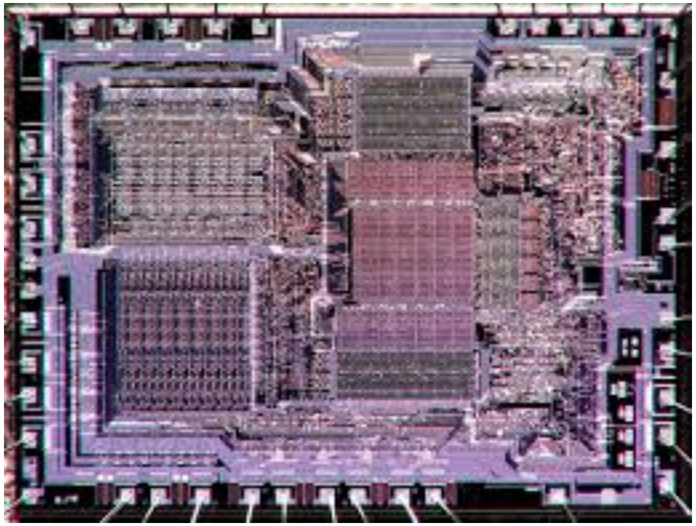
```
load R1, A
mul R1, B               /*     A*B              */
load R2, C
mul R2, B               /*     C*B              */
add R2, A               /*     A + CB                 */
sub R2, R1              /*     AB - (A + C*B)         */
```

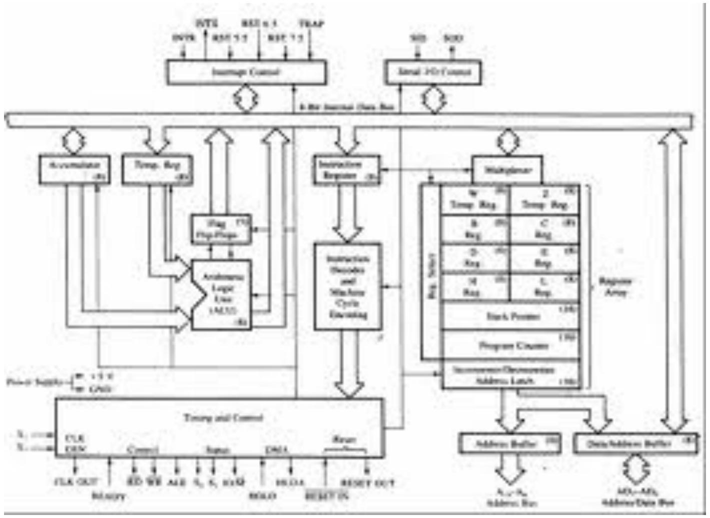
Memory-Register: Pros and Cons

- Pros
  - Some data can be accessed without loading first
  - Instruction format easy to encode
  - Good code density

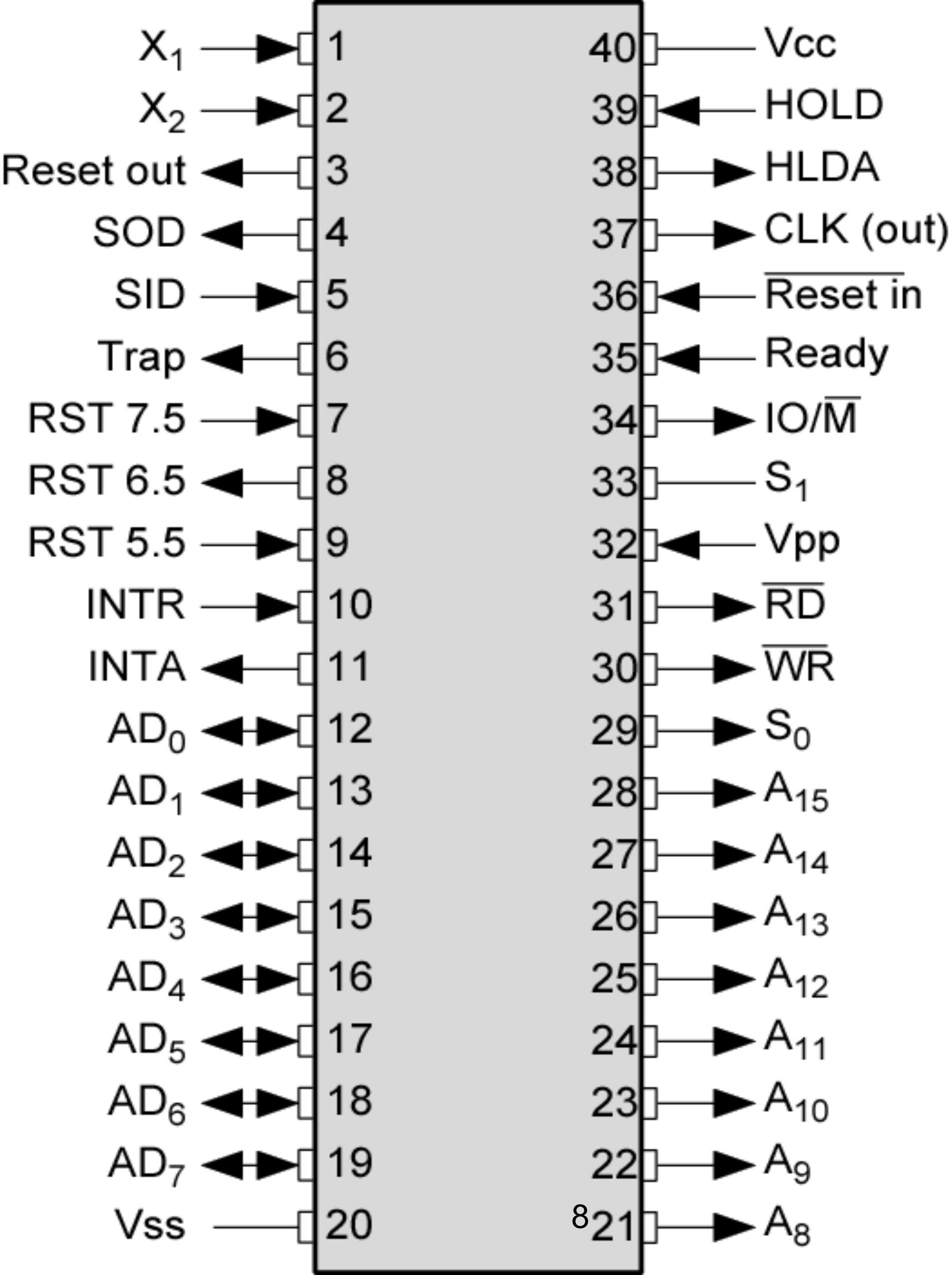
- Cons
  - Operands are not equivalent (poor orthogonality)
  - Variable number of clocks per instruction
  - May limit number of registers

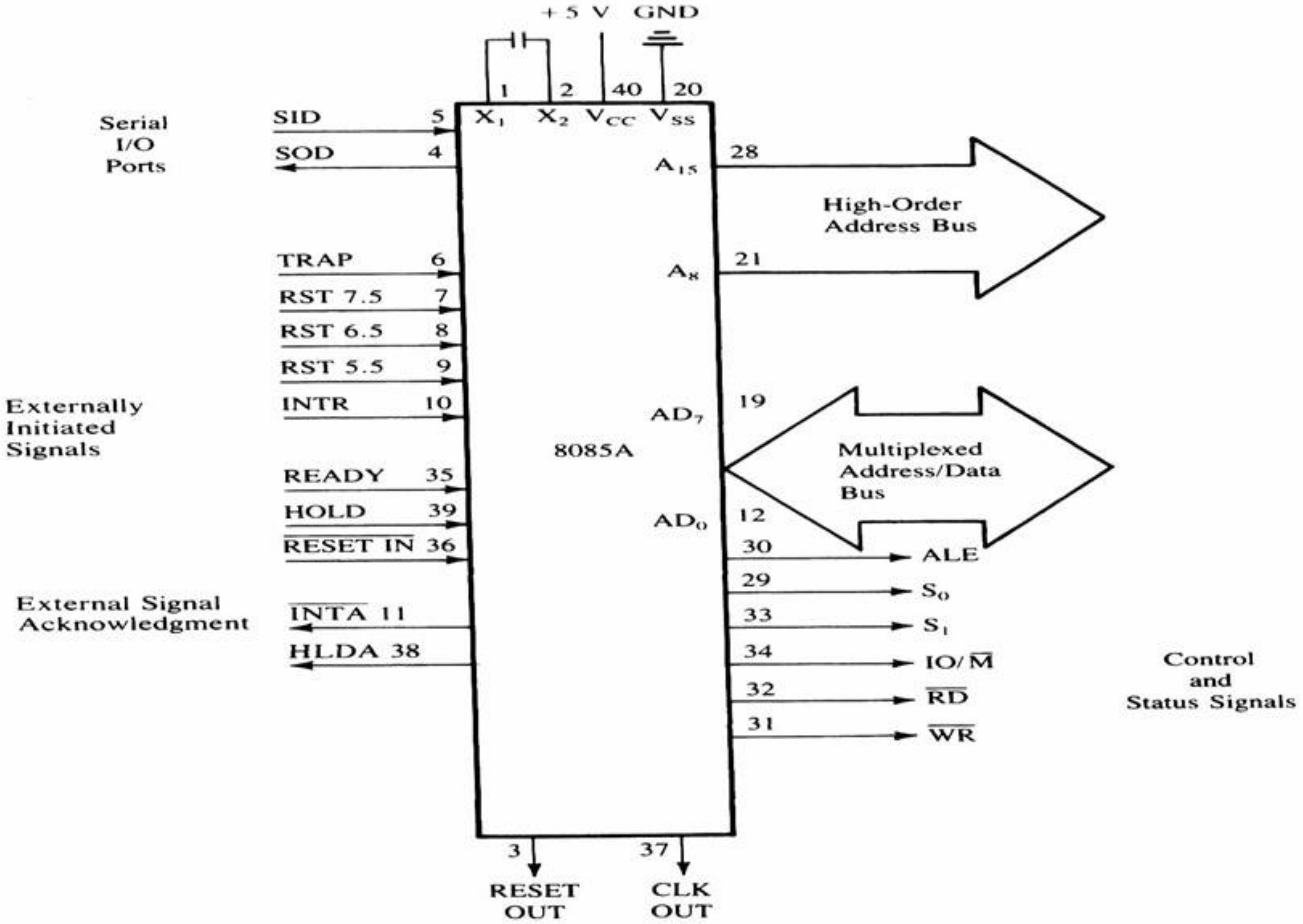
Microprocessor: 8085





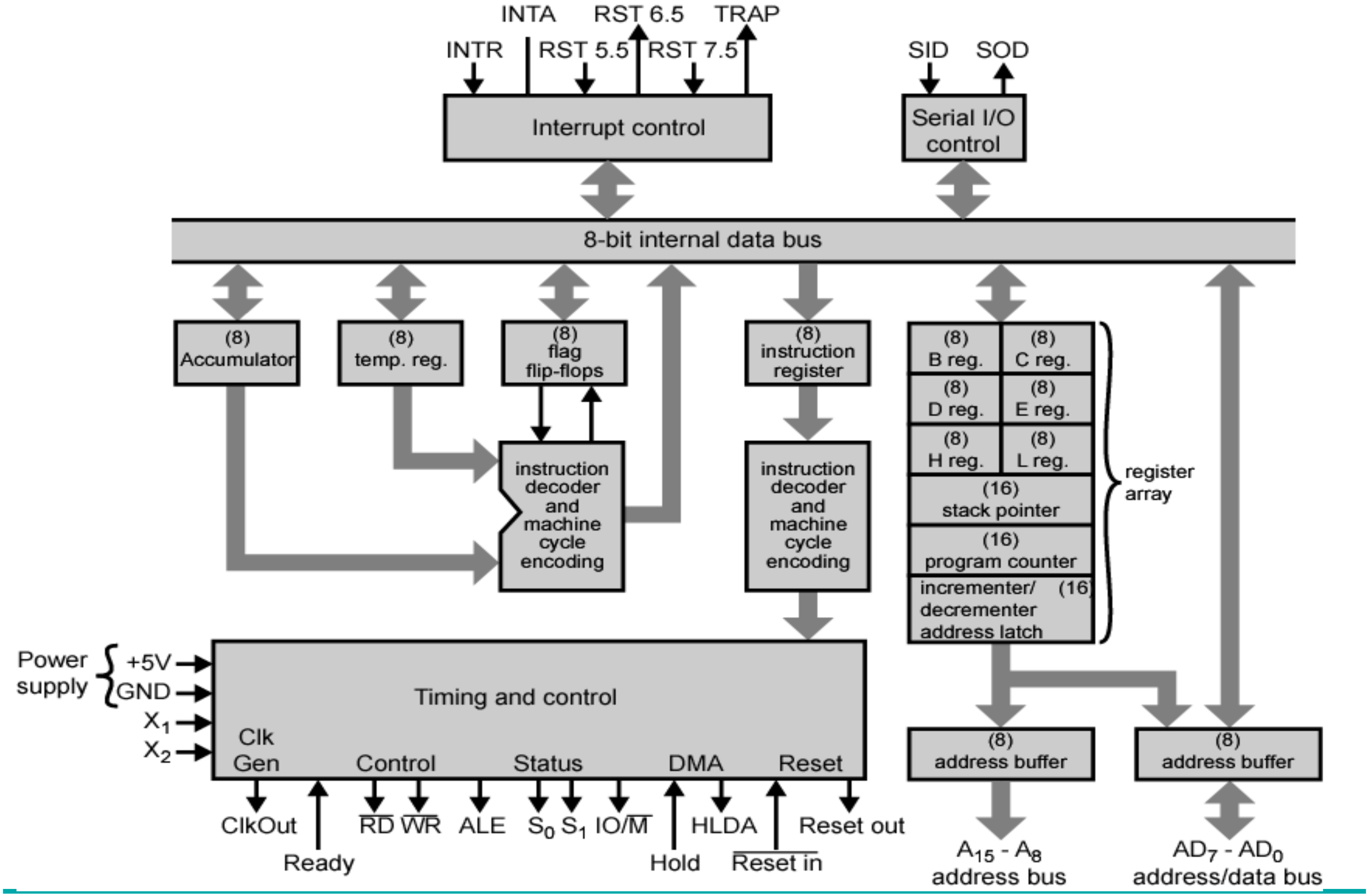
Intel 8085 Pin Configuration





Signals and I/O Pins

Intel 8085 CPU Block Diagram



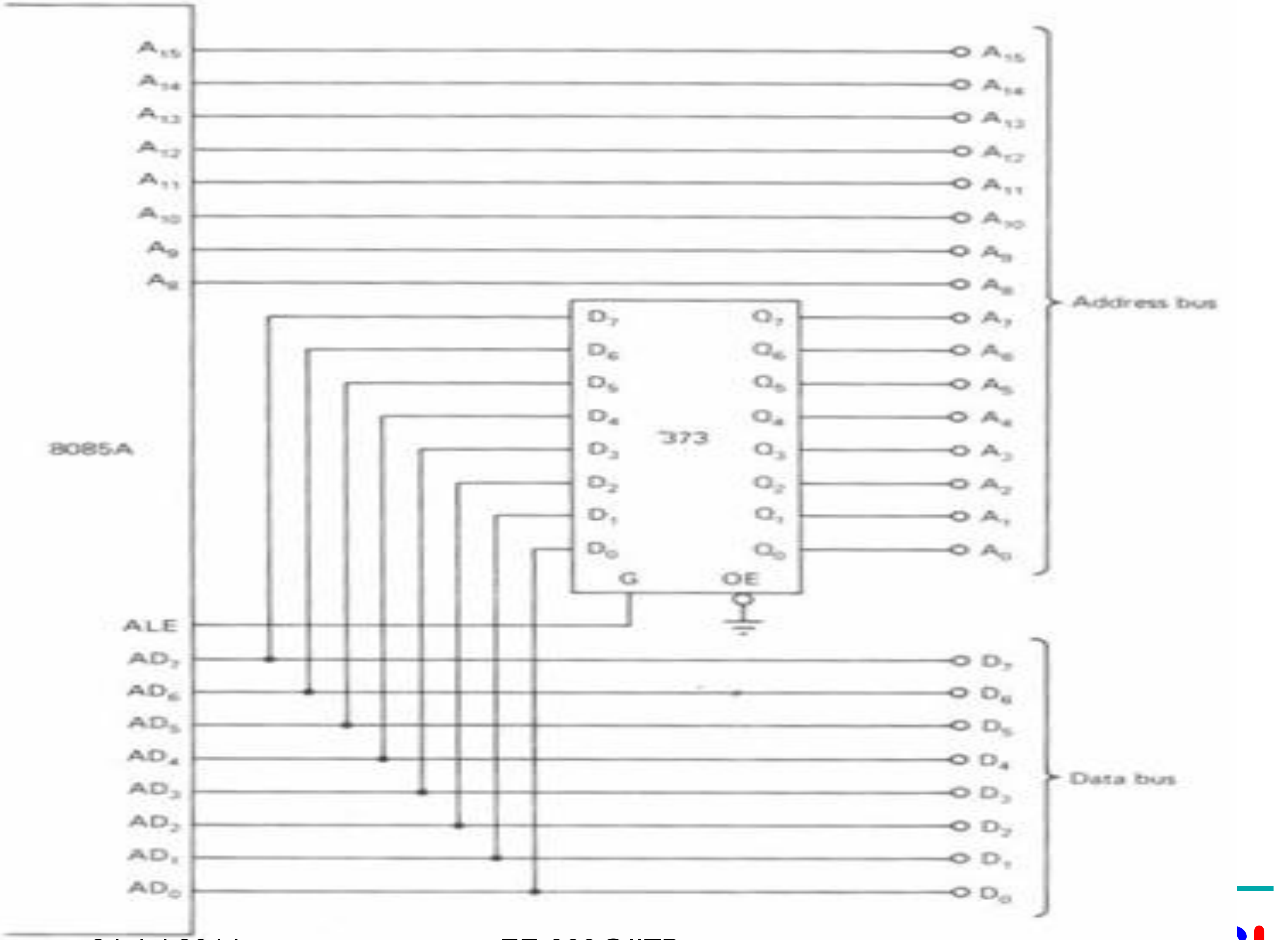
The 8085 and Its Buses

- 8-bit general purpose microprocessor that can address 64K Byte of memory.
- 40 pins and uses +5V for power. It can run at a maximum frequency of 3 MHz.
  - The pins on the chip can be grouped into 6 groups
    - Address Bus.
    - Data Bus.
    - Control and Status Signals.
    - Power supply and frequency.
    - Externally Initiated Signals.
    - Serial I/O ports.

The Address and Data Bus Systems

- The address bus has 8 signal lines A8 – A15 which are unidirectional.
- The other 8 address bits are multiplexed (time shared) with the 8 data bits.
  - So, the bits AD0 – AD7 are bi-directional and serve as A0 – A7 and D0 – D7 at the same time.
    - During the execution of the instruction, these lines carry the address bits during the early part, then during the late parts of the execution, they carry the 8 data bits.
  - In order to separate the address from the data, we can use a latch to save the value before the function of the bits changes.

Demultiplexing of Address/Data Bus

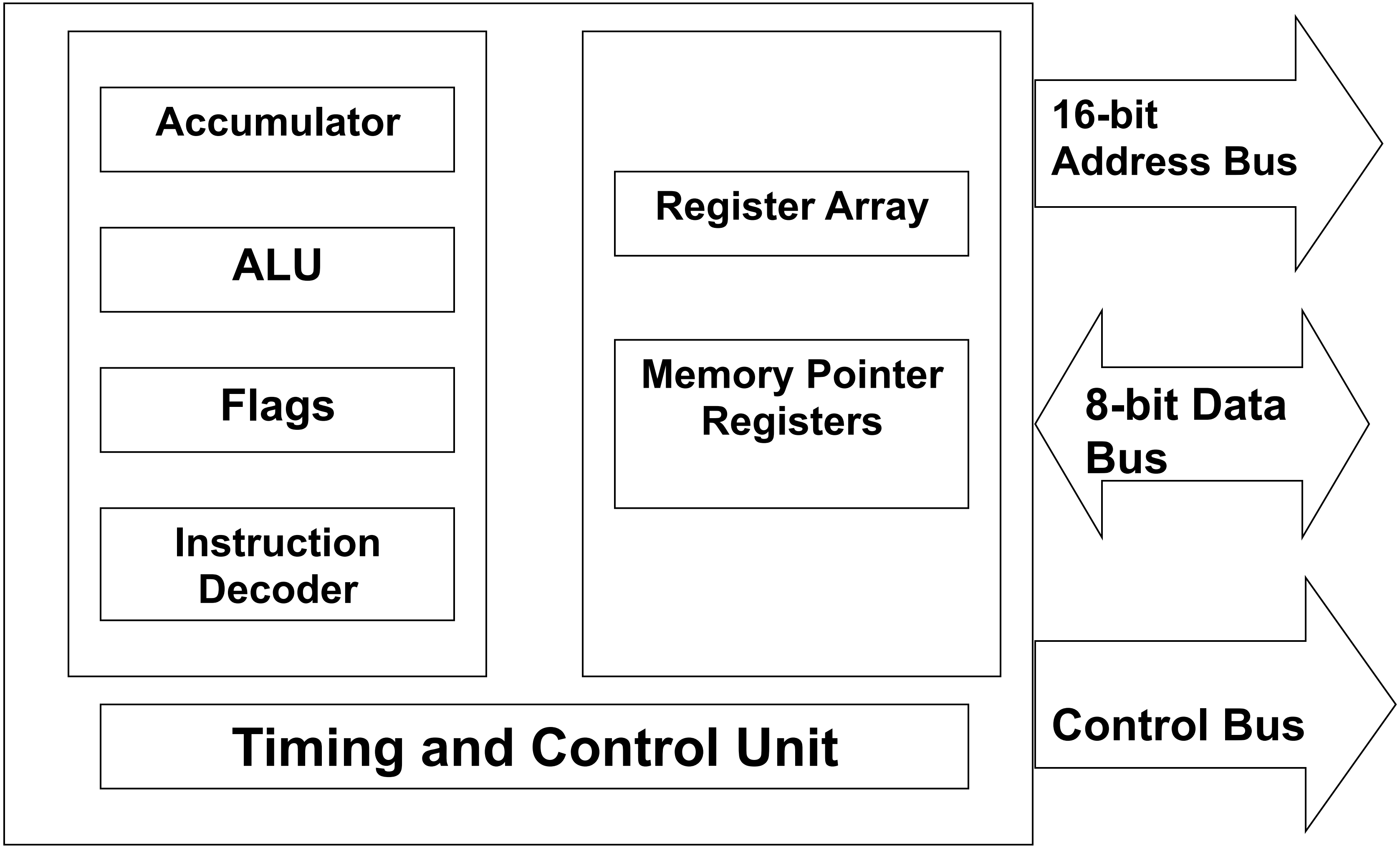


The Control and Status Signals

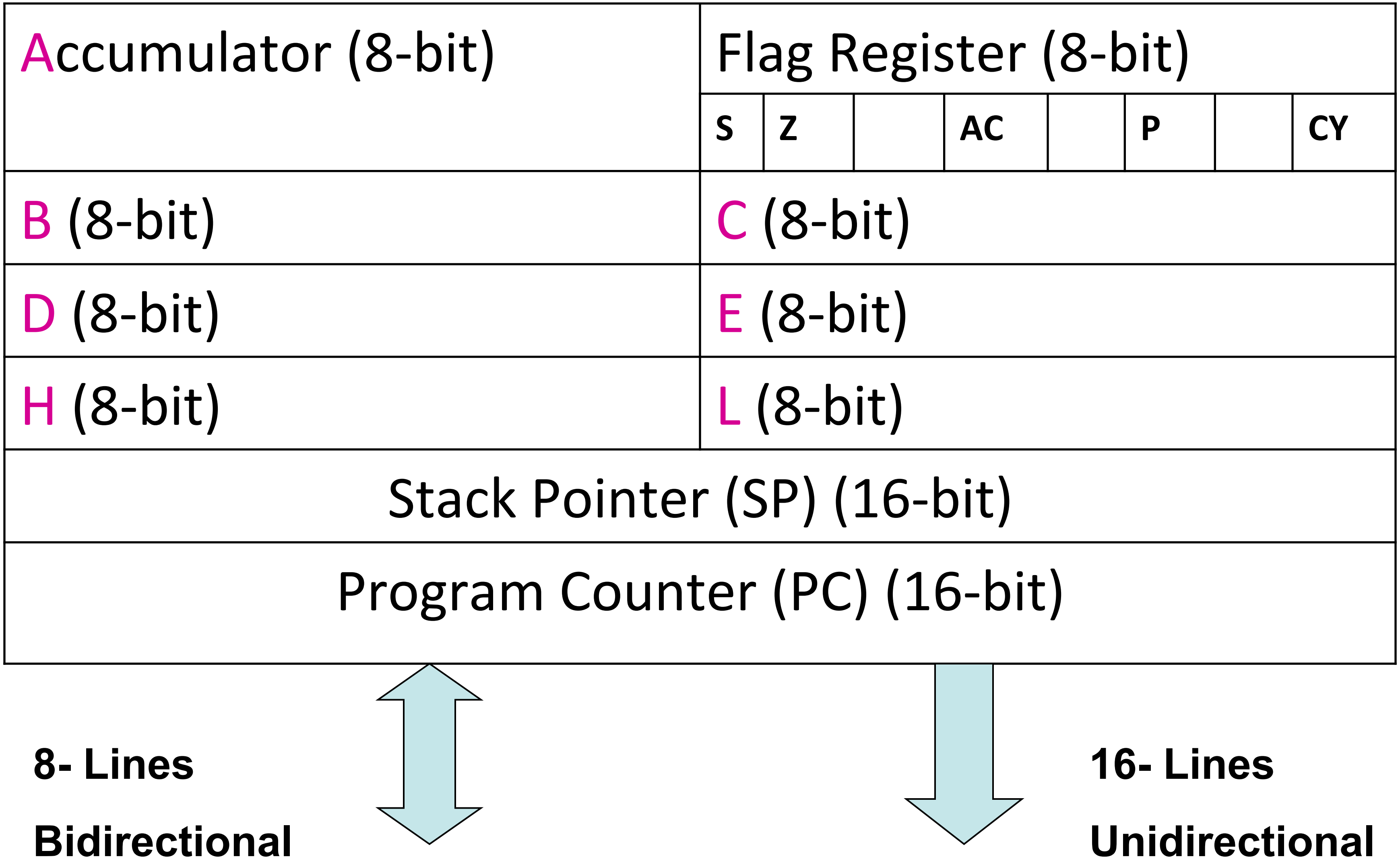
- 4 main control and status signals.
  - ALE: Address Latch Enable. This signal is a pulse that become 1 when the AD0 – AD7 lines have an address on them. It becomes 0 after that. This signal can be used to enable a latch to save the address bits from the AD lines.
  - RD: Read. Active low.
  - WR: Write. Active low.
  - IO/M: This signal specifies whether the operation is a memory operation (IO/M=0) or an I/O operation (IO/M=1).
  - S1 and S0 : Status signals to specify the kind of operation being performed. Usually not used in small systems.

INSTRUCTION SET OF 8085

Programming model of 8085



Programming model of 8085



<table>
<tr><td rowspan="2">Accumulator (8-bit)</td><td colspan="8">Flag Register (8-bit)</td></tr>
<tr><td>S</td><td>Z</td><td></td><td>AC</td><td></td><td>P</td><td></td><td>CY</td></tr>
<tr><td>B (8-bit)</td><td colspan="8">C (8-bit)</td></tr>
<tr><td>D (8-bit)</td><td colspan="8">E (8-bit)</td></tr>
<tr><td>H (8-bit)</td><td colspan="8">L (8-bit)</td></tr>
<tr><td colspan="9">Stack Pointer (SP) (16-bit)</td></tr>
<tr><td colspan="9">Program Counter (PC) (16-bit)</td></tr>
</table>

8- Lines

Bidirectional

16- Lines

Unidirectional

Data movement instructions for the 8085 microprocessor

Instruction	Operation
NOP	No operation
MOV *r*1, *r*2	*r*1 = *r*2
MOV *r*, *M*	*r* = M[*HL*]
MOV *M*, *r*	M[*HL*] = *r*
MVI *r*, *n*	*r* = *n*
MVI *M*, *n*	M[*HL*] = *n*
LXI *rp*, Γ	*rp* = Γ
LDA Γ	*A* = M[Γ]
STA Γ	M[Γ] = *A*
LHLD Γ	*HL* = M[Γ], M[Γ+1]
SHDL Γ	M[Γ], M[Γ+1] = *HL*

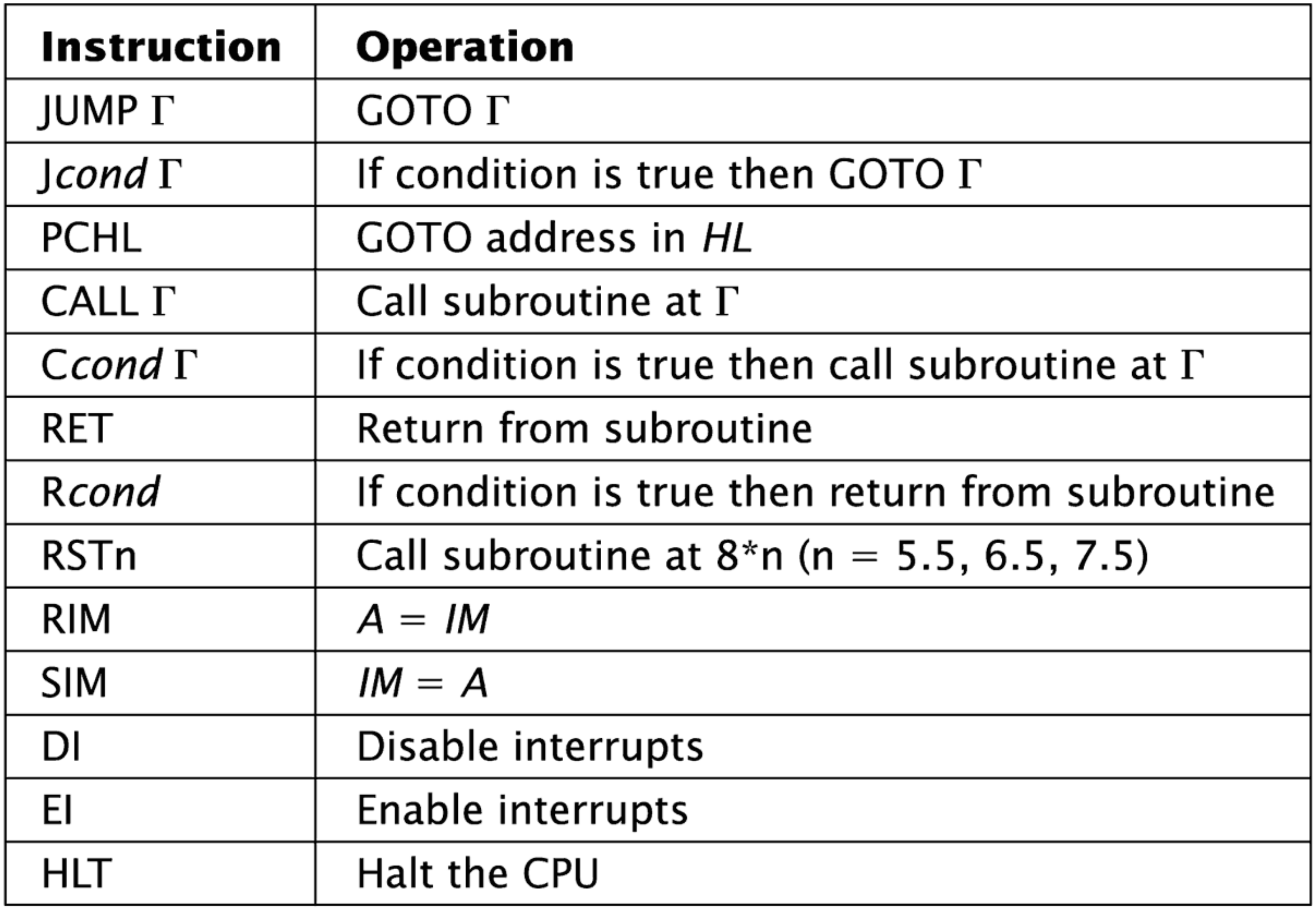
Instruction	Operation
LDAX *rp*	*A* = M[*rp*] (*rp* = *BC*, *DE*)
STAX *rp*	M[*rp*] = *A* (*rp* = *BC*, *DE*)
XCHG	*DE* ↔ *HL*
PUSH *rp*	Stack = *rp* (*rp* ≠ *SP*)
PUSH PSW	Stack = *A*, flag register
POP *rp*	*rp* = Stack (*rp* ≠ *SP*)
POP PSW	*A*, flag register = Stack
XTHL	*HL* ↔ Stack
SPHL	*SP* = *HL*
IN *n*	*A* = input port *n*
OUT *n*	output port *n* = *A*

Data Operation instructions for the 8085 microprocessor

Instruction	Operation	Flags
ADD *r*	$A = A + r$	All
ADD *M*	$A = A + \mathrm{M}[HL]$	All
ADI *n*	$A = A + n$	All
ADC *r*	$A = A + r + CY$	All
ADC *M*	$A = A + \mathrm{M}[HL] + CY$	All
ACI *n*	$A = A + n + CY$	All
SUB *r*	$A = A - r$	All
SUB *M*	$A = A - \mathrm{M}[HL]$	All
SUI *n*	$A = A - n$	All
SBB *r*	$A = A - r - CY$	All
SBB *M*	$A = A - \mathrm{M}[HL] - CY$	All
SBI *n*	$A = A - n - CY$	All
INR *r*	$r = r + 1$	Not *CY*
INR *M*	$\mathrm{M}[HL] = \mathrm{M}[HL] + 1$	Not *CY*
DCR *n*	$r = r - 1$	Not *CY*
DCR *M*	$\mathrm{M}[HL] = \mathrm{M}[HL] - 1$	Not *CY*
INX *rp*	$rp = rp + 1$	None
DCX *rp*	$rp = rp - 1$	None
DAD *rp*	$HL = HL + rp$	*CY*
DAA	Decimal adjust	All

Instruction	Operation	Flags
ANA *r*	$A = A \wedge r$	All
ANA *M*	$A = A \wedge \mathrm{M}[HL]$	All
ANI *n*	$A = A \wedge n$	All
ORA *r*	$A = A \vee r$	All
ORA *M*	$A = A \vee \mathrm{M}[HL]$	All
ORI *n*	$A = A \vee n$	All
XRA *r*	$A = A \oplus r$	All
XRA *M*	$A = A \oplus \mathrm{M}[HL]$	All
XRI *n*	$A = A \oplus n$	All
CMP *r*	Compare *A* and *r*	All
CMP *M*	Compare *A* and M[*HL*]	All
CPI *n*	Compare *A* and *n*	All
RLC	$CY = \mathrm{A}_7, A = \mathrm{A}_{6-0}, \mathrm{A}_7$	*CY*
RRC	$CY = \mathrm{A}_0, A = \mathrm{A}_0, \mathrm{A}_{7-1}$	*CY*
RAL	$CY, A = A, CY$	*CY*
RAR	$A, CY = CY, A$	*CY*
CMA	$A = A'$	None
CMC	$CY = CY'$	*CY*
STC	$CY = 1$	*CY*

Program Control instructions for the 8085 microprocessor



Instruction	**Operation**
JUMP Γ	GOTO Γ
J*cond* Γ	If condition is true then GOTO Γ
PCHL	GOTO address in *HL*
CALL Γ	Call subroutine at Γ
C*cond* Γ	If condition is true then call subroutine at Γ
RET	Return from subroutine
R*cond*	If condition is true then return from subroutine
RSTn	Call subroutine at 8*n (n = 5.5, 6.5, 7.5)
RIM	$A = IM$
SIM	$IM = A$
DI	Disable interrupts
EI	Enable interrupts
HLT	Halt the CPU

Instruction Set of 8085

- Instructions
  - 74 operation codes, e.g. ADD
  - 246 Instructions, e.g. ADD B
- 8085 instructions can be classified as
  1. **Data Transfer (Copy)**
  2. **Arithmetic**
  3. **Logical and Bit manipulation**
  4. **Branch**
  5. **Machine Control**

Addressing Modes of 8085

- The various formats of specifying operands are called addressing modes
- Addressing modes of 8085
  - Register Addressing
  - Immediate Addressing
  - Memory Addressing
  - Input/Output Addressing

Register Addressing

- Operands are one of the internal registers of 8085
- Examples-

  MOV A, B

  ADD C

Immediate Addressing

- Value of the operand is given in the instruction itself
- Example-

MVI A, 20H

LXI H, 2050H

ADI 30H

SUI 10H

Memory Addressing

- One of the operands is a memory location
- Depending on how address of memory location is specified, **memory** addressing is of two types
  - **Direct** addressing
  - **Indirect** addressing

Direct Addressing

- 16-bit Address of the memory location is specified in the instruction directly
- Examples-

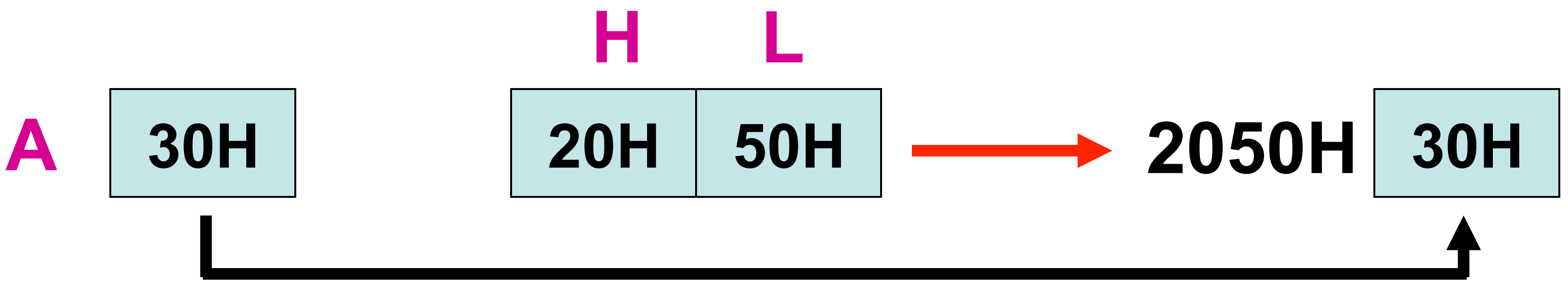
LDA 2050H

STA 3050H

Indirect Addressing

- A memory pointer register is used to store the address of the memory location
- Example-

MOV M, A ;copy register A to memory location whose address is stored in register pair HL



Input/Output Addressing

- 8-bit address of the port is directly specified in the instruction
- Examples-

  IN 07H

  OUT 21H

Instruction & Data Formats

Instruction classification according to size

- 1-byte Instructions
- 2-byte Instructions
- 3-byte Instructions

One-byte Instructions

- Includes Opcode and Operand in the same byte
- Examples-

Opcode	Operand	Binary Code	Hex Code
MOV	C, A	0100 1111	**4F**H
ADD	B	1000 0000	**80**H
HLT		0111 0110	**76**H

Two-byte Instructions

- First byte specifies Operation Code
- Second byte specifies Operand

Opcode	Operand	Binary Code	Hex Code
MVI	A, 32H	0011 1110	3EH
		0011 0010	32H
MVI	B, F2H	0000 0110	06H
		1111 0010	F2H

Three-byte Instructions

- First byte specifies Operation Code
- Second & Third byte specifies Operand

Opcode	Operand	Binary Code	Hex Code
LXI	H, 2050H	0010 0001	21H
		0101 0000	50H
		0010 0000	20H
LDA	3070H	0011 1010	3AH
		0111 0000	70H
		0011 0000	30H

Data Transfer (Copy) Operations

1. **Load** a 8-bit number in a Register
2. **Copy** from Register to Register
3. **Copy** between Register and Memory
4. **Copy** between Input/Output Port and Accumulator
5. **Load** a 16-bit number in a Register pair
6. **Copy** between Register pair and Stack memory

Example Data Transfer (Copy) Operations / Instructions

1. **Load** a 8-bit number 4F in register B	**MVI B, 4FH**
2. **Copy** from Register B to Register A	**MOV A,B**
3. **Load** a 16-bit number 2050 in Register pair HL	**LXI H, 2050H**
4. **Copy** from Register B to Memory Address 2050	**MOV M,B**
5. **Copy** between Input/Output Port and Accumulator	**OUT 01H** **IN 07H**

Arithmetic Operations

1. **Addition** of two 8-bit numbers
2. **Subtraction** of two 8-bit numbers
3. **Increment/ Decrement** a 8-bit number

Example Arithmetic Operations / Instructions

1. **Add** a 8-bit number 32H to Accumulator	ADI 32H
2. **Add** contents of Register B to Accumulator	ADD B
3. **Subtract** a 8-bit number 32H from Accumulator	SUI 32H
4. **Subtract** contents of Register C from Accumulator	SUB C
5. **Increment** the contents of Register D by 1	INR D
6. **Decrement** the contents of Register E by 1	DCR E

Logical & Bit Manipulation Operations

- **AND** two 8-bit numbers
- **OR** two 8-bit numbers
- **Exclusive-OR** two 8-bit numbers
- **Compare** two 8-bit numbers
- **Complement**
- **Rotate** Left/Right Accumulator bits

Example Logical & Bit Manipulation Operations / Instructions

○ Logically **AND** Register H with Accumulator	**ANA H**
○ Logically **OR** Register L with Accumulator	**ORA L**
○ Logically **XOR** Register B with Accumulator	**XRA B**
○ **Compare** contents of Register C with Accumulator	**CMP C**
○ **Complement** Accumulator	**CMA**
○ **Rotate** Accumulator Left	**RAL**

Branching Operations

- control the flow of program execution

  - **Jumps**
    - Conditional jumps
    - Unconditional jumps
  - **Call & Return**
    - Conditional Call & Return
    - Unconditional Call & Return

Example Branching Operations / Instructions

1. **Jump** to a 16-bit Address 2080H if Carry flag is **SET**	**JC 2080H**
2. Unconditional **Jump**	**JMP 2050H**
3. **Call** a subroutine with its 16-bit Address	**CALL 3050H**
4. **Return back** from the Call	**RET**
5. **Call** a subroutine with its 16-bit Address if **C**arry flag is **RESET**	**CNC 3050H**
6. **Return** if **Z**ero flag is **SET**	**RZ**

Machine Control Instructions

- Affect the operation of the processor

HLT	Stop program execution
NOP	Do not perform any operation

Writing a Assembly Language Program

Steps to write a program

- – Analyze the problem
- – Develop algorithm
- – Draw a flowchart
- – Convert flowchart to Assembly language by picking appropriate 8085 instructions

Example 1

- Add two numbers
- Copy 8-bit result in A to C
- If CARRY is generated
  - Handle it
- Result is in register pair BC

Algorithm Development

1. Load two numbers in registers D, E

3. Add them

5. Store 8 bit result in C

7. Check CARRY flag

8. If CARRY flag is SET
    - Store CARRY in register B

9. Stop

- Load registers D, E

- Copy register D to A
- Add register E to A

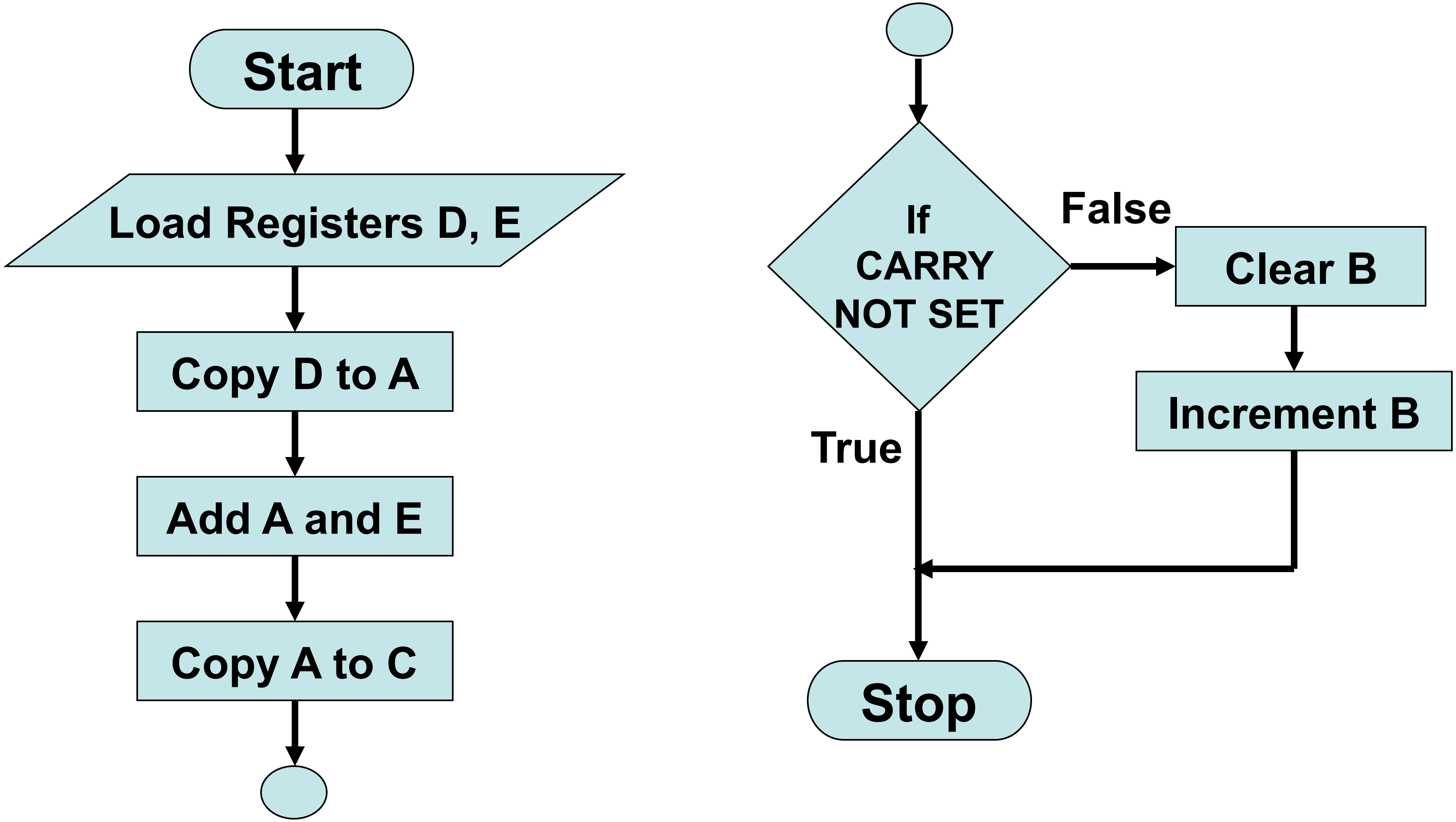
- Copy A to register C

- Use Conditional Jump instructions

- Clear register B
- Increment B

- Stop processing

Flowchart


Start
Load Registers D, E
Copy D to A
Add A and E
Copy A to C
If CARRY NOT SET
False
Clear B
Increment B
True
Stop

Assembly Language Program

- Load registers D, E
- Copy register D to A
- Add register E to A
- Copy A to register C
- Use Conditional Jump instructions
- Clear register B
- Increment B
- Stop processing

```
      MVI D, 2H
      MVI E, 3H
      MOV A, D
      ADD E
      MOV C, A
      JNC END
      MVI B, 0H
      INR B
END:  HLT
```

Example 2: Sum of Natural Numbers

The Algorithm of the program

```
1:      total = 0, i = 0
2:      i = i + 1
3:      total = total + i
4:      IF I /= n THEN GOTO 2
```

n + (n - 1) + … + 1

The 8085 coding of the program

```
        LDA  n          } i = n
        MOV  B, A       }
        XRA  A          } sum = A XOR A = 0
Loop:   ADD  B          } sum = sum + i
        DCR  B          } i = i - 1
        JNZ  Loop       } IF I /= 0 THEN GOTO Loop
        STA  total      } total = sum
```

Thank You