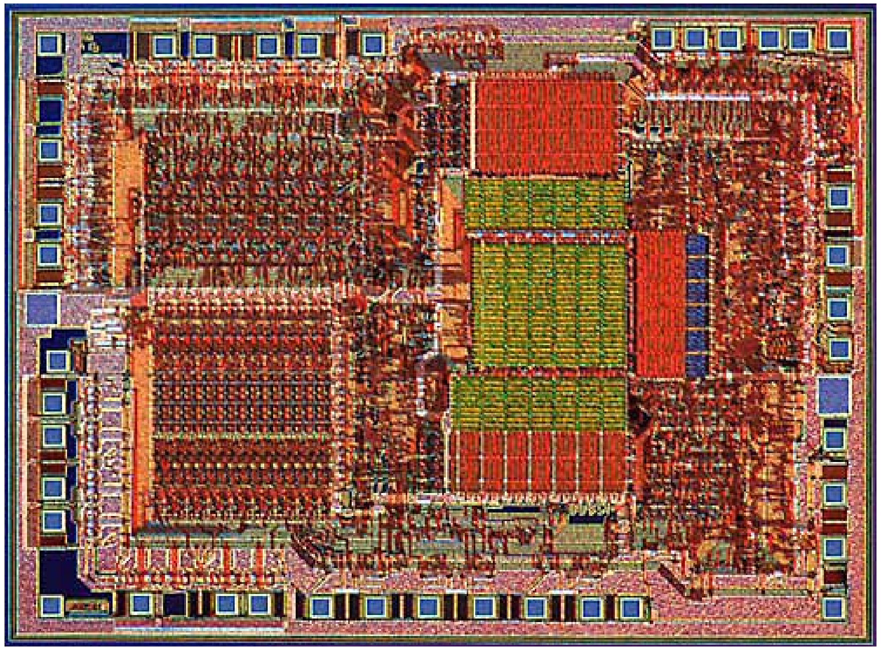
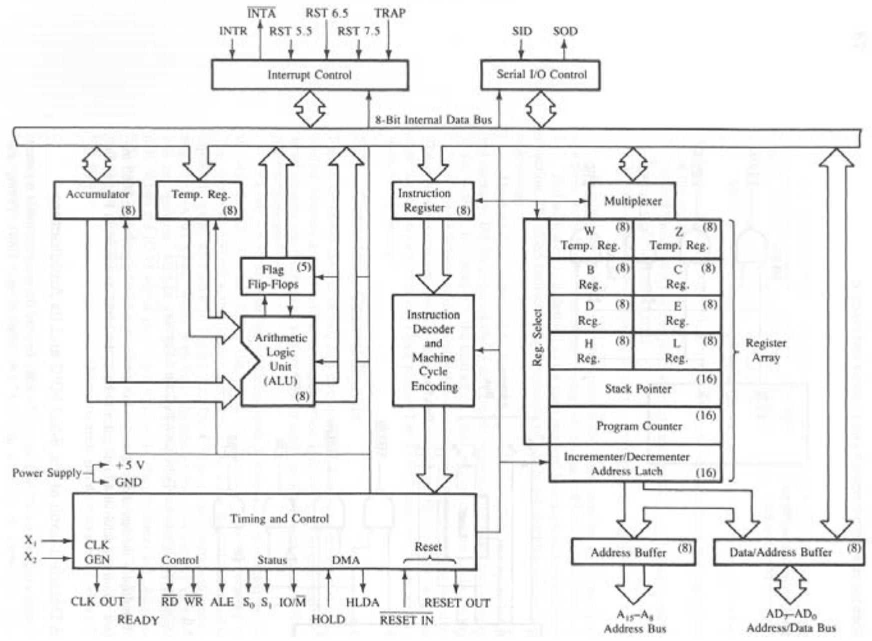
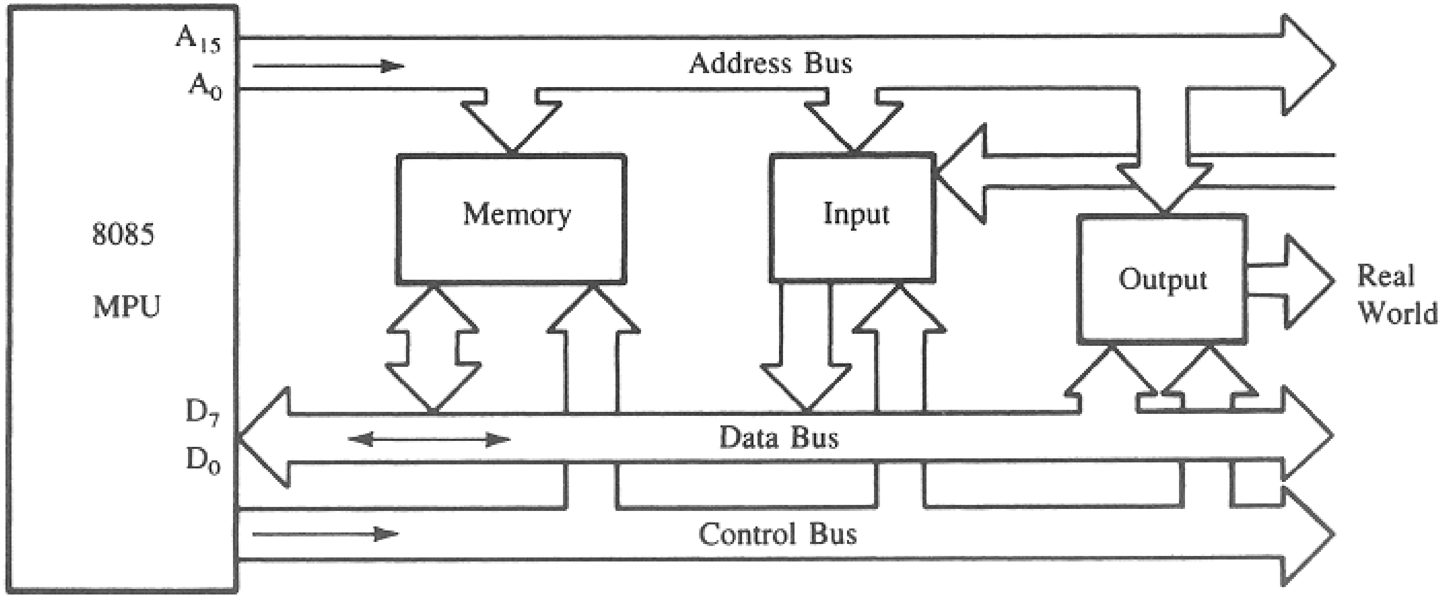
Intel 8085 Microprocessor Architecture





The 8085 Bus Structure

The 8-bit 8085 CPU (or MPU – Micro Processing Unit) communicates with the other units using a 16-bit address bus, an 8-bit data bus and a control bus.



The 8085 Bus Structure

Address Bus

- Consists of 16 address lines: $A_0 – A_{15}$
- Operates in **unidirectional** mode: The address bits are always sent from the MPU to peripheral devices, not reverse.
- 16 address lines are capable of addressing a total of 2^{16} = 65,536 (64k) memory locations.
- Address locations: 0000 (hex) – FFFF (hex)

The 8085 Bus Structure

Data Bus

- Consists of 8 data lines: $D_0 – D_7$
- Operates in **bidirectional** mode: The data bits are sent from the MPU to peripheral devices, as well as from the peripheral devices to the MPU.
- Data range: 00 (hex) – FF (hex)

Control Bus

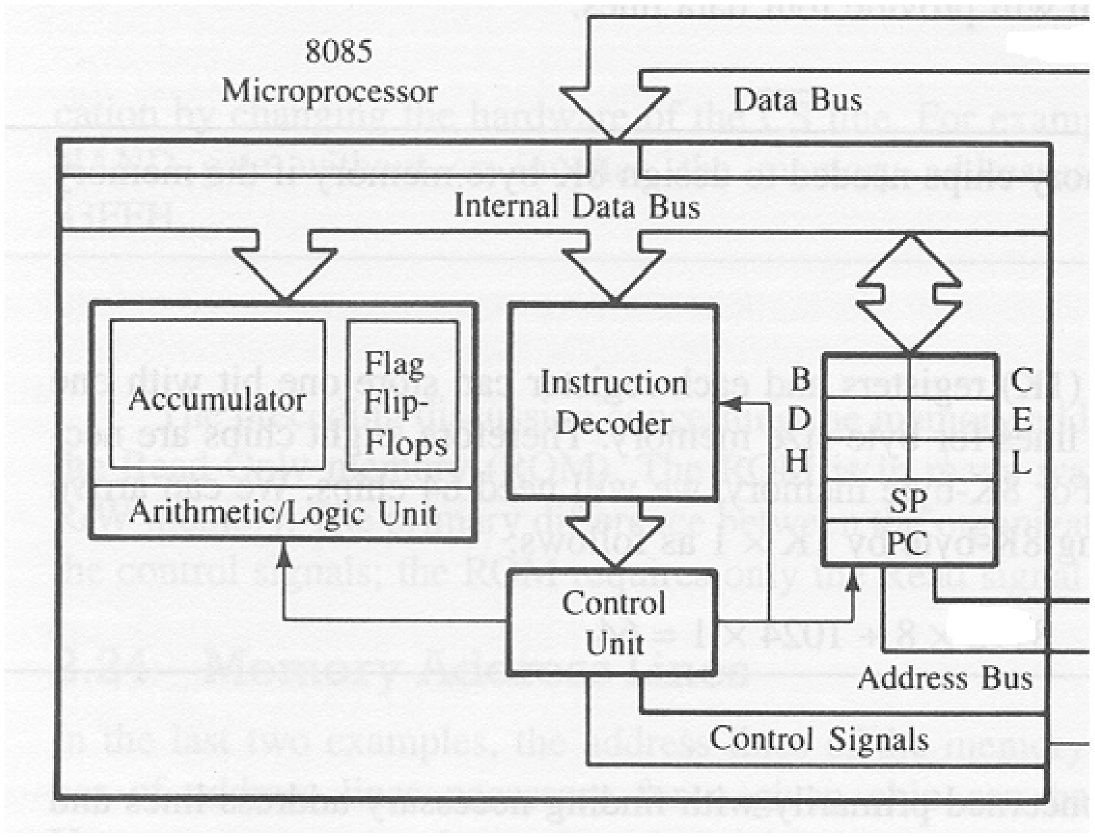
- Consists of various lines carrying the control signals such as read / write enable, flag bits.

The 8085: CPU Internal Structure

The internal architecture of the 8085 CPU is capable of performing the following operations:

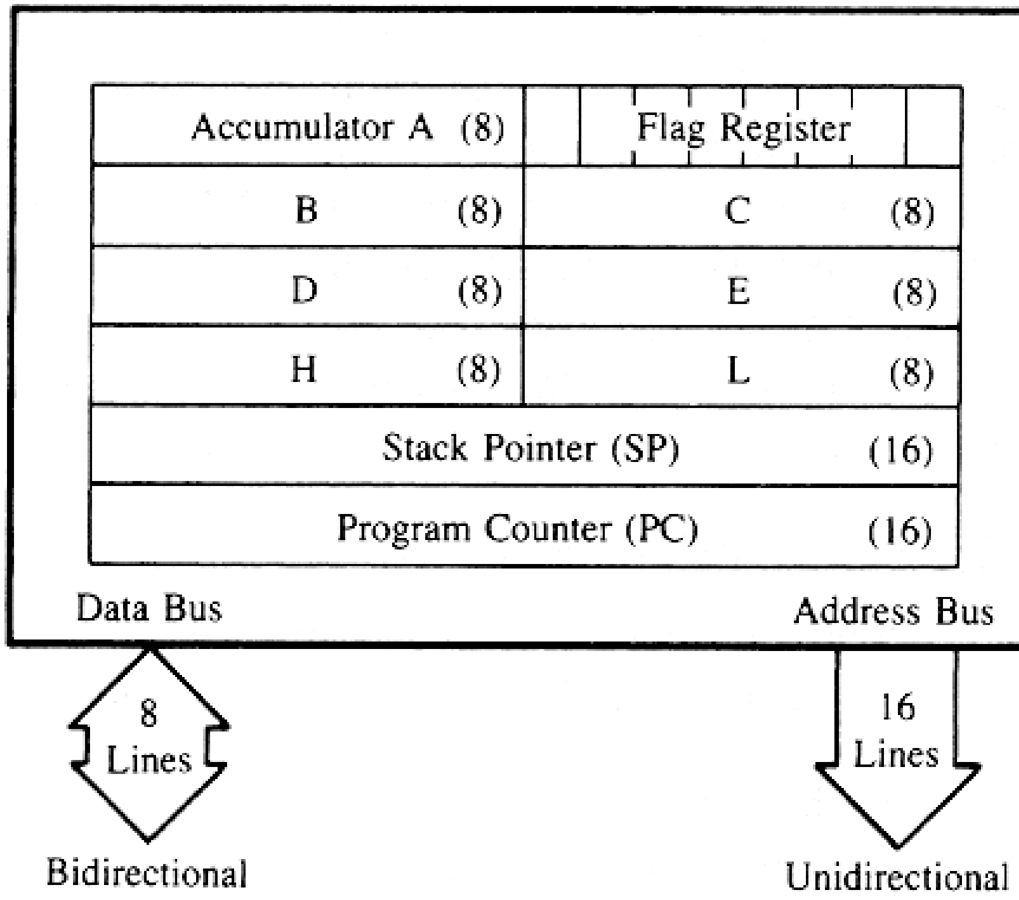
- Store 8-bit data (Registers, Accumulator)
- Perform arithmetic and logic operations (ALU)
- Test for conditions (IF / THEN)
- Sequence the execution of instructions
- Store temporary data in RAM during execution

The 8085: CPU Internal Structure



Simplified block diagram

The 8085: Registers



The 8085: CPU Internal Structure

Registers

- Six general purpose 8-bit registers: B, C, D, E, H, L
- They can also be combined as register pairs to perform 16-bit operations: BC, DE, HL
- Registers are programmable (data load, move, etc.)

Accumulator

- Single 8-bit register that is part of the ALU !
- Used for arithmetic / logic operations – the result is always stored in the accumulator.

The 8085: CPU Internal Structure

Flag Bits

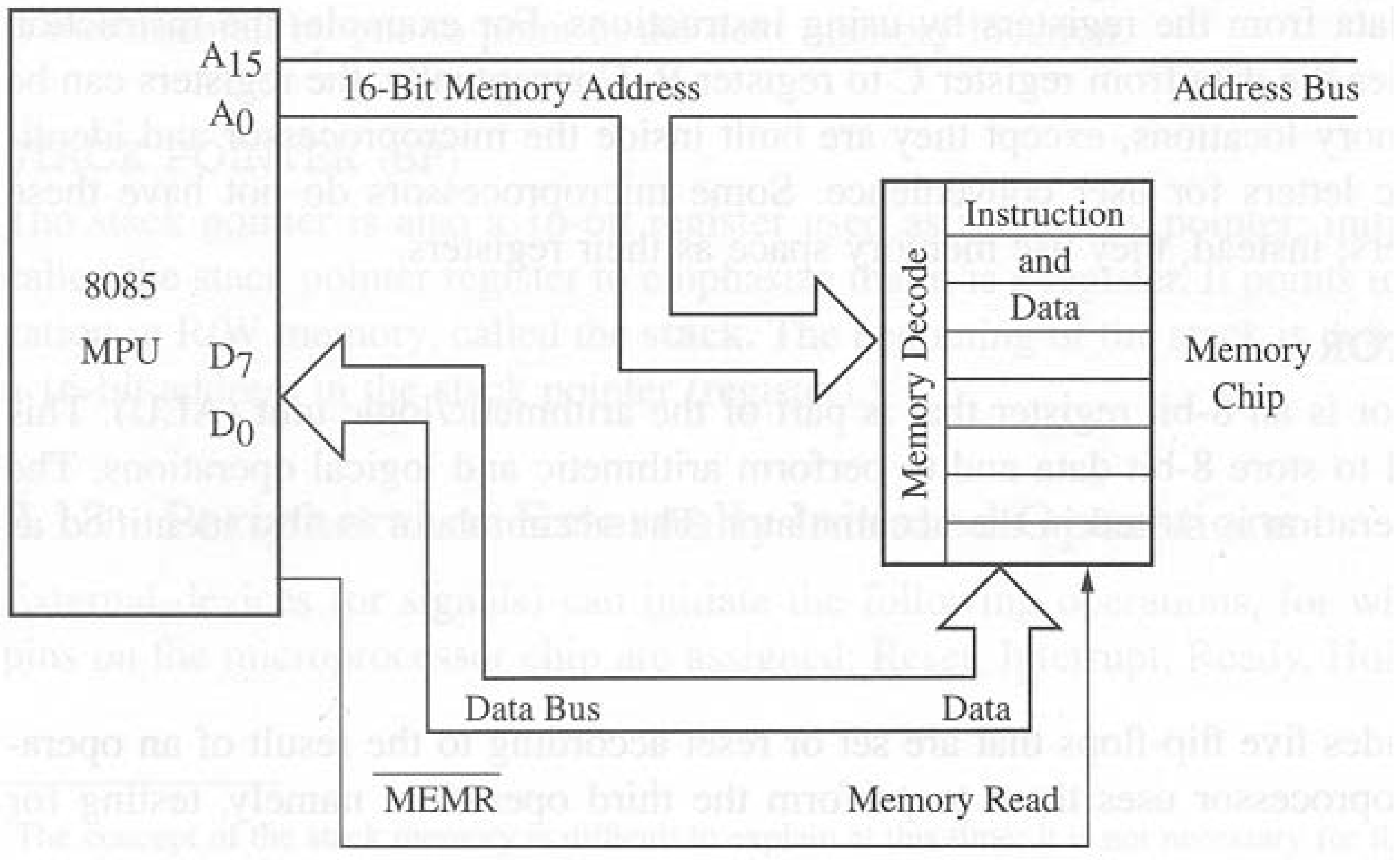
- Indicate the result of condition tests.
- Carry, Zero, Sign, Parity, etc.
- Conditional operations (IF / THEN) are executed based on the condition of these flag bits.

Program Counter (PC)

- Contains the memory address (16 bits) of the instruction that will be executed in the next step.

Stack Pointer (SP)

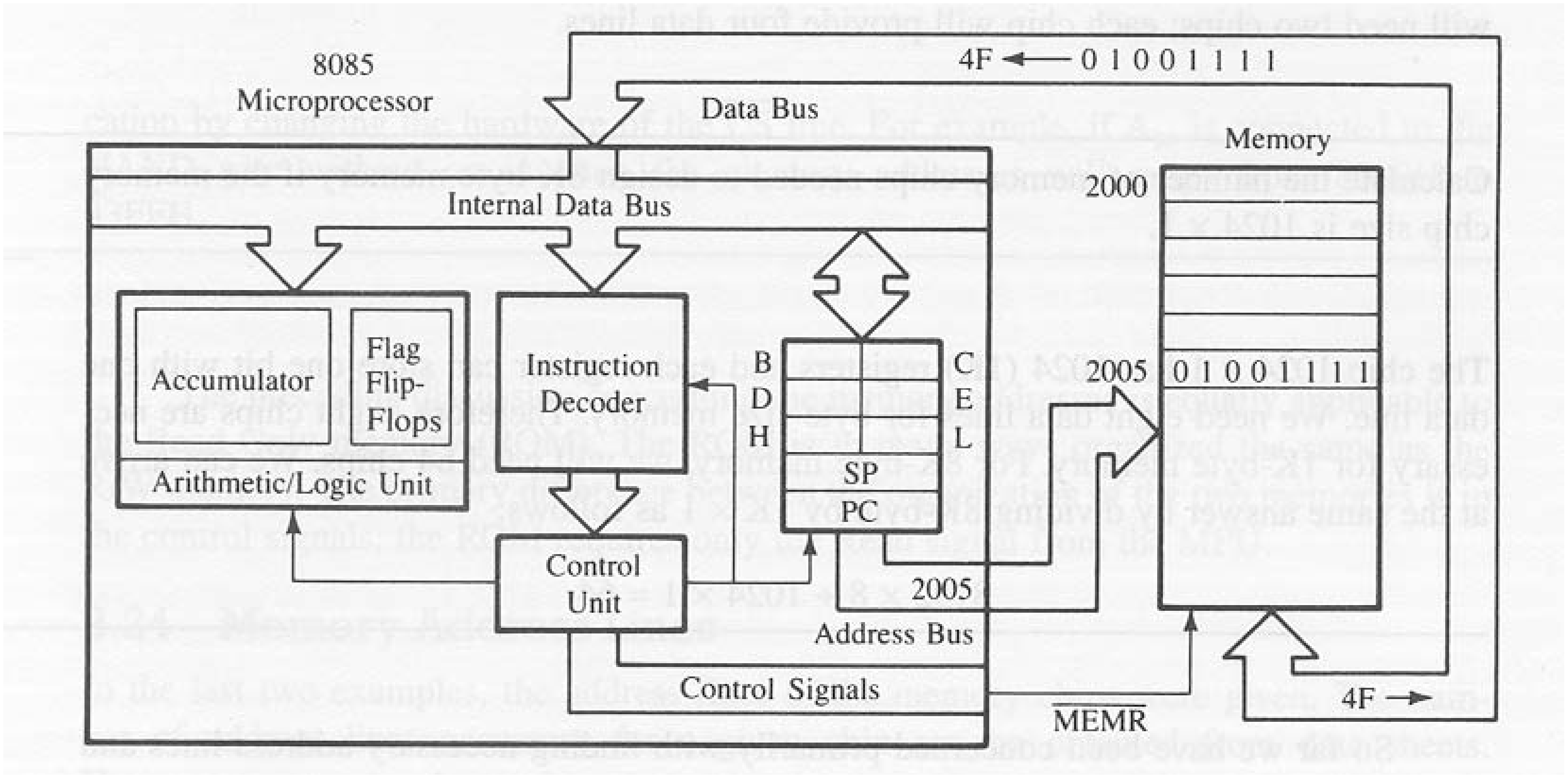
Example: Memory Read Operation



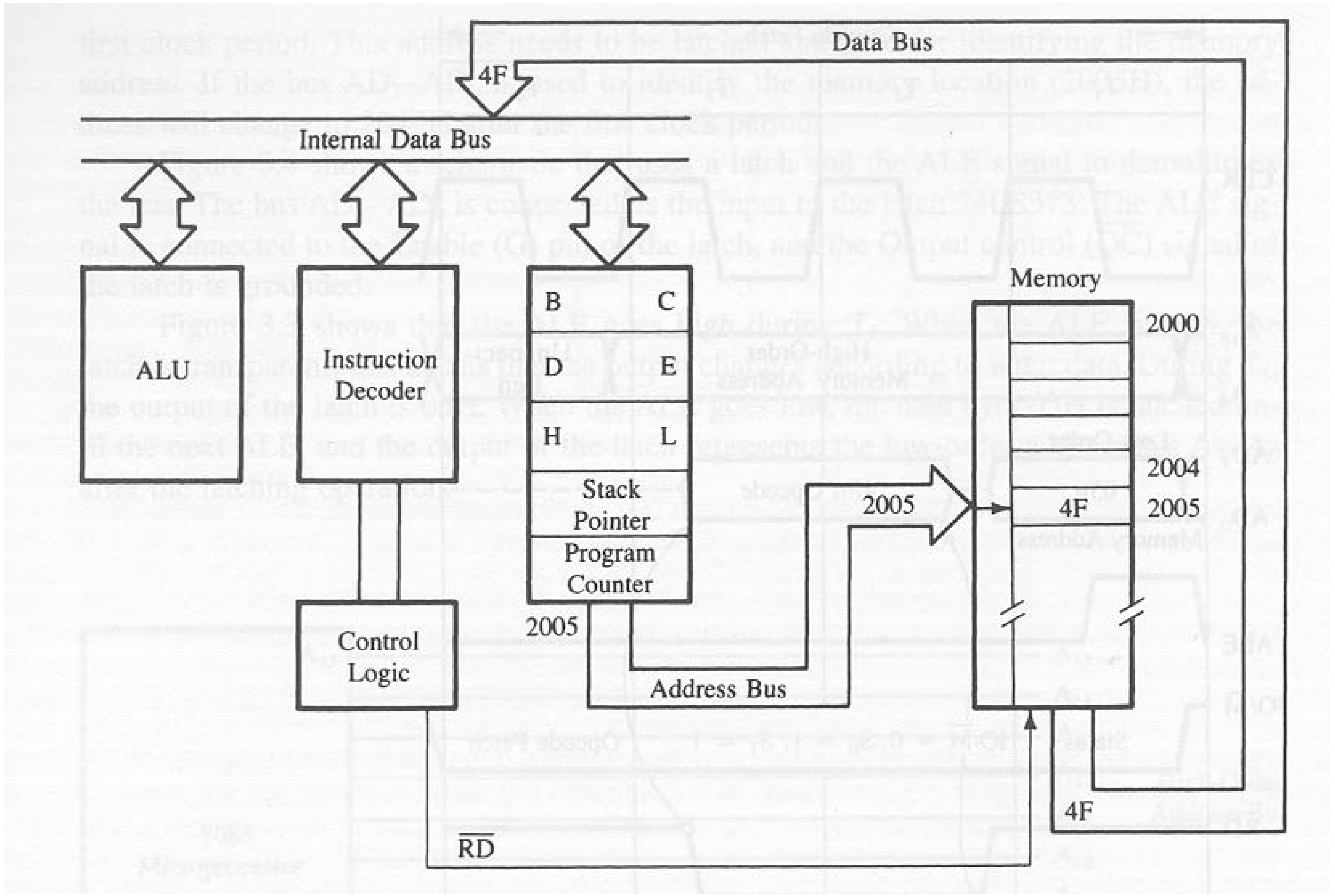
Example: Instruction Fetch Operation

- All instructions (program steps) are stored in memory.
- To run a program, the individual instructions must be read from the memory in sequence, and executed.
  - Program counter puts the 16-bit memory address of the instruction on the address bus
  - Control unit sends the Memory Read Enable signal to access the memory
  - The 8-bit instruction stored in memory is placed on the data bus and transferred to the instruction decoder
  - Instruction is decoded and executed

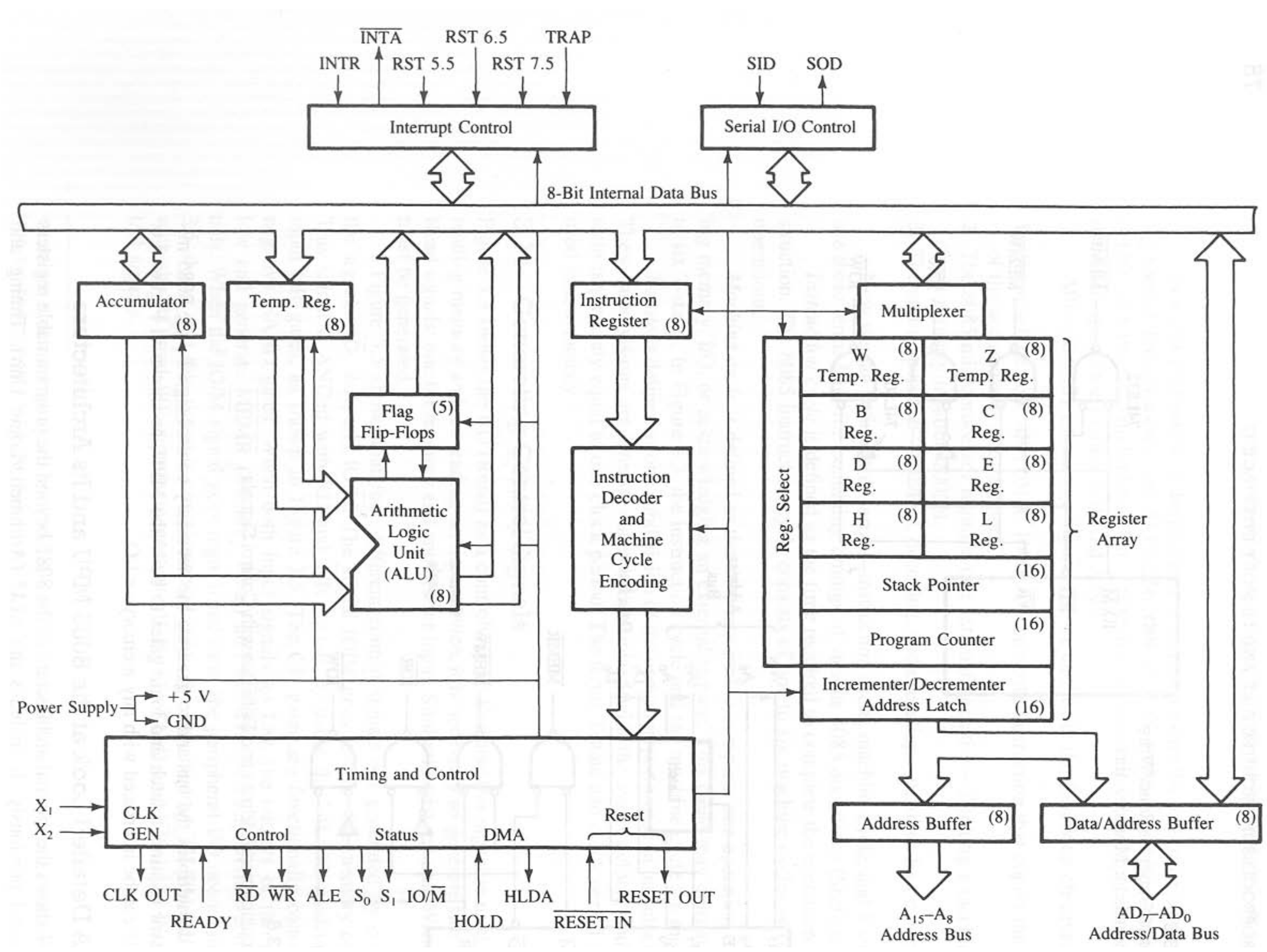
Example: Instruction Fetch Operation



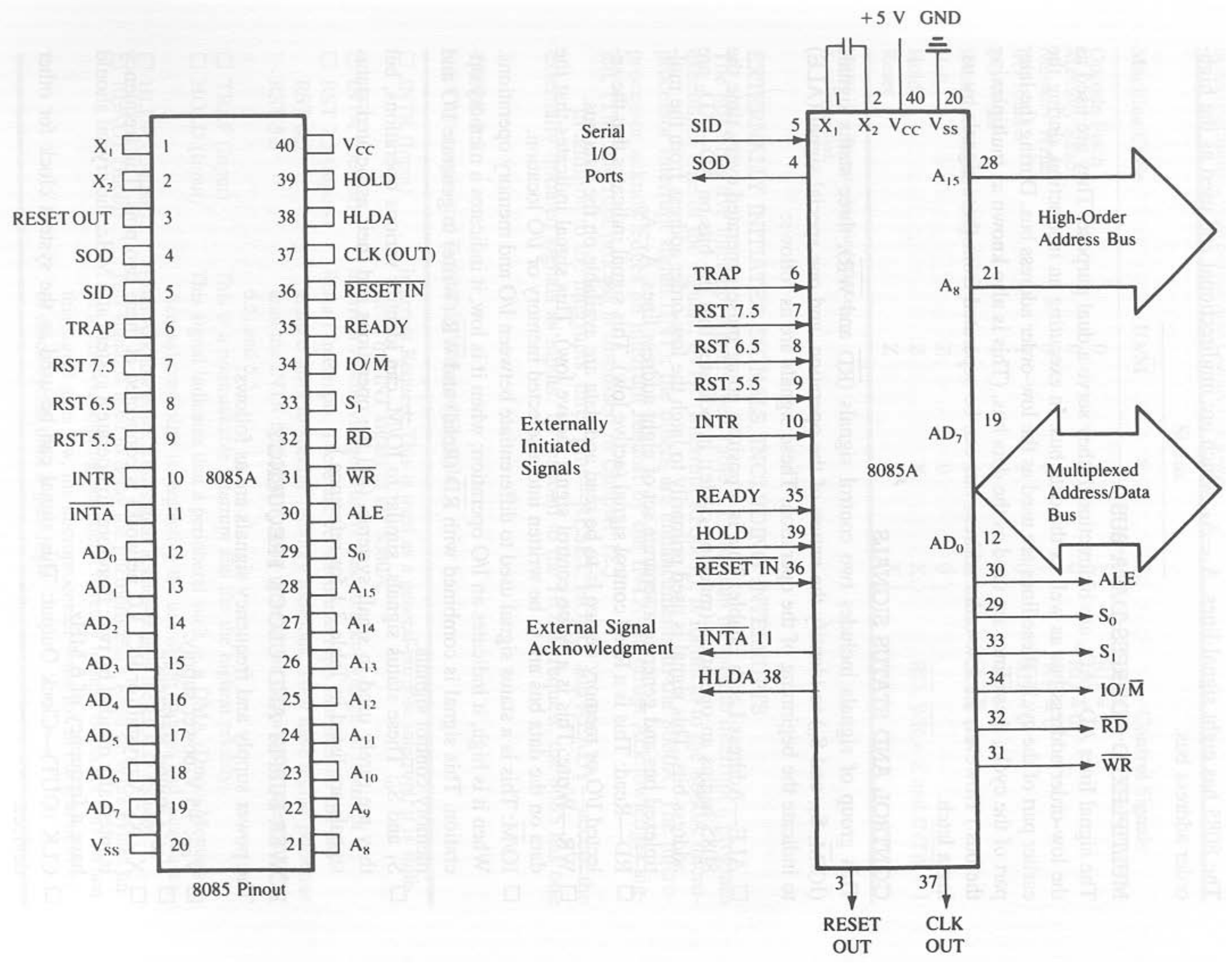
Example: Instruction Fetch Operation



8085 Functional Block Diagram



The 8085 Microprocessor



X1 1, X2 2, RESET OUT 3, SOD 4, SID 5, TRAP 6, RST 7.5 7, RST 6.5 8, RST 5.5 9, INTR 10, INTA 11, AD0 12, AD1 13, AD2 14, AD3 15, AD4 16, AD5 17, AD6 18, AD7 19, VSS 20
40 VCC, 39 HOLD, 38 HLDA, 37 CLK (OUT), 36 RESET IN, 35 READY, 34 IO/M, 33 S1, 32 RD, 31 WR, 30 ALE, 29 S0, 28 A15, 27 A14, 26 A13, 25 A12, 24 A11, 23 A10, 22 A9, 21 A8
8085A
8085 Pinout
+5 V
GND
X1 X2 VCC VSS
Serial I/O Ports
SID 5
SOD 4
Externally Initiated Signals
TRAP 6
RST 7.5 7
RST 6.5 8
RST 5.5 9
INTR 10
READY 35
HOLD 39
RESET IN 36
External Signal Acknowledgment
INTA 11
HLDA 38
A15 28
A8 21
High-Order Address Bus
AD7 19
AD0 12
Multiplexed Address/Data Bus
30 ALE
29 S0
33 S1
34 IO/M
32 RD
31 WR
3 RESET OUT
37 CLK OUT

Basic Processor Operation

- All instructions (program steps) are stored in memory.
- To run a program, the individual instructions must be read from the memory in sequence, and executed.
- Detailed sequence:
  - Instruction fetch (read from memory)
  - Decode instruction
  - Get operands
  - Execute operation
  - Save result

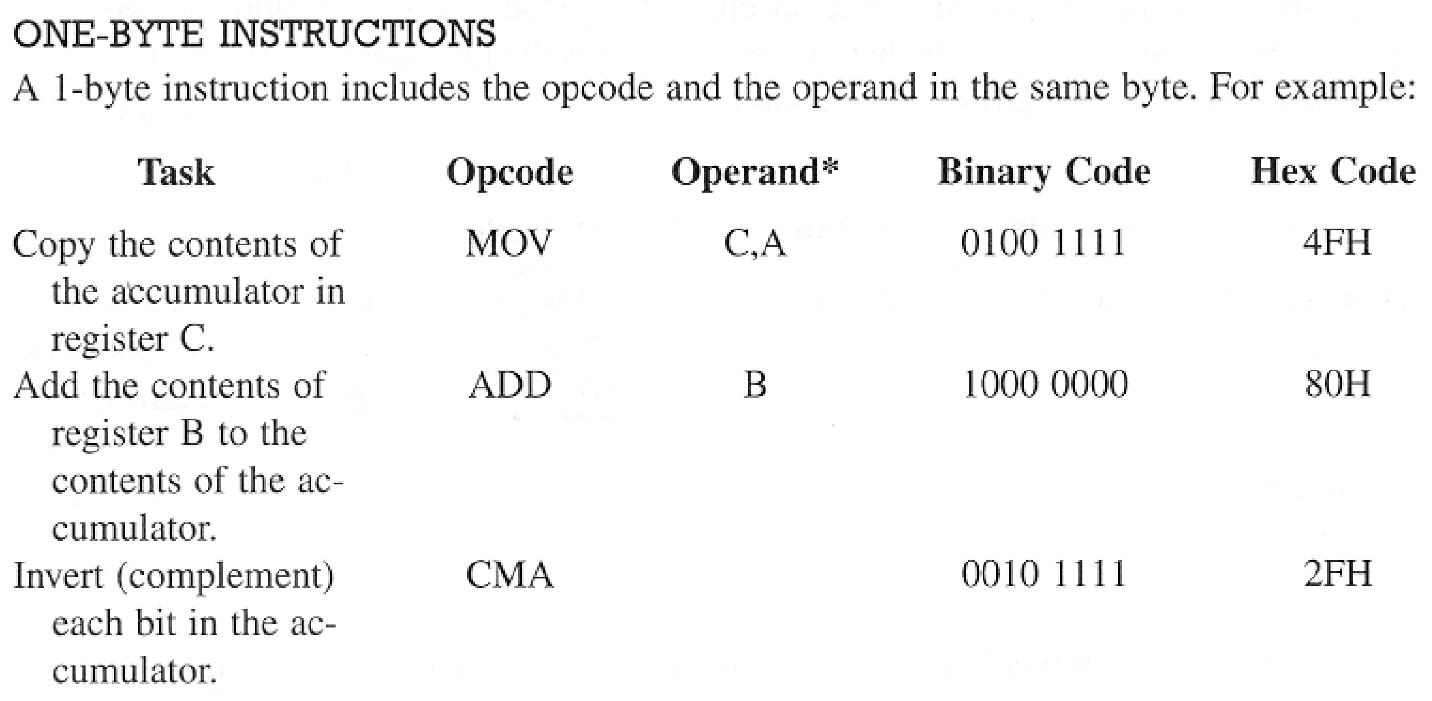
get next instruction

8085 Instruction Set

The 8085 instructions can be classified as follows:

- Data transfer operations
  - Between registers
  - Between memory location and a register
  - Direct write to a register / memory
  - Between I/O device and accumulator
- Arithmetic operations (ADD, SUB, INR, DCR)
- Logic operations
- Branching operations (JMP, CALL, RET)

8085 Instruction Types



ONE-BYTE INSTRUCTIONS

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

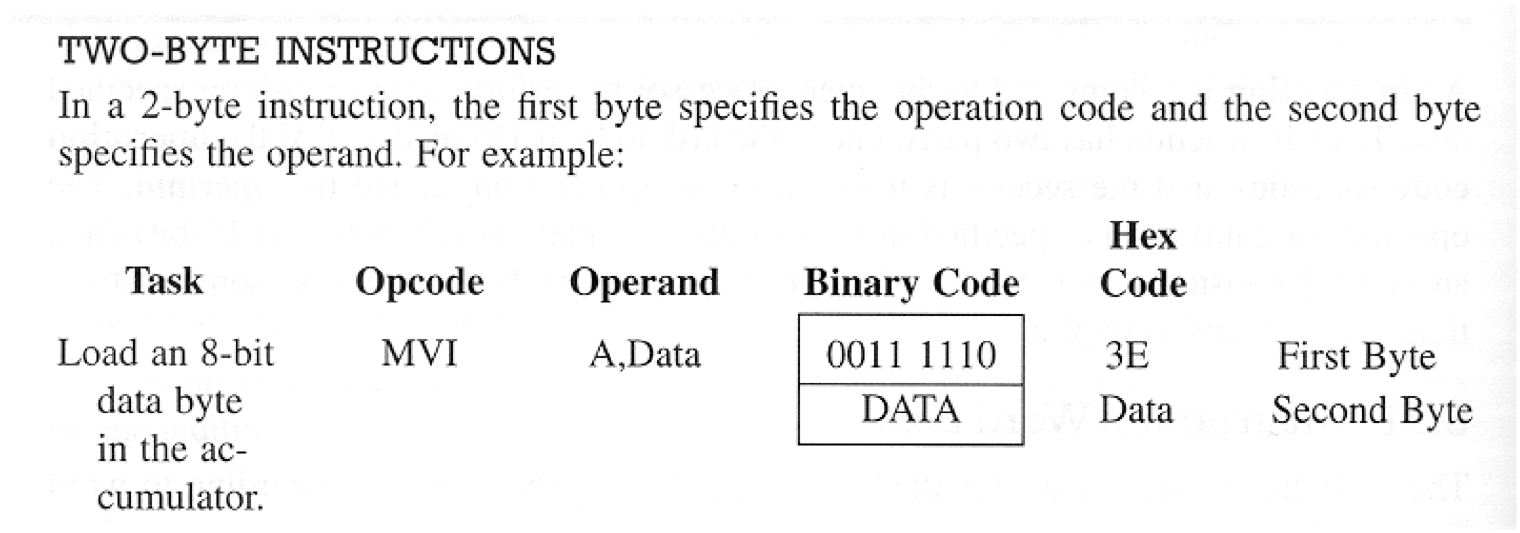
1. MOV and CMA: No FLAGS affected.
2. ADD affects the FLAGS.

8085 Flags Register

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

1. **Z – Zero: The Zero flag is set to 1 when the result is zero; otherwise it is reset.**
2. **CY – Carry: If an arithmetic operation results in a carry, the CY flag is set; otherwise it is reset.**
3. **S – Sign: The Sign flag is set if bit D_7 of the result = 1; otherwise it is reset.**
4. **P – Parity: If the result has an even number of 1s, the flag is set; for an odd number of 1s, the flag is reset;**
5. **AC – Auxilary Carry: Check the manual. We will not use it in our labs.**

8085 Instruction Types



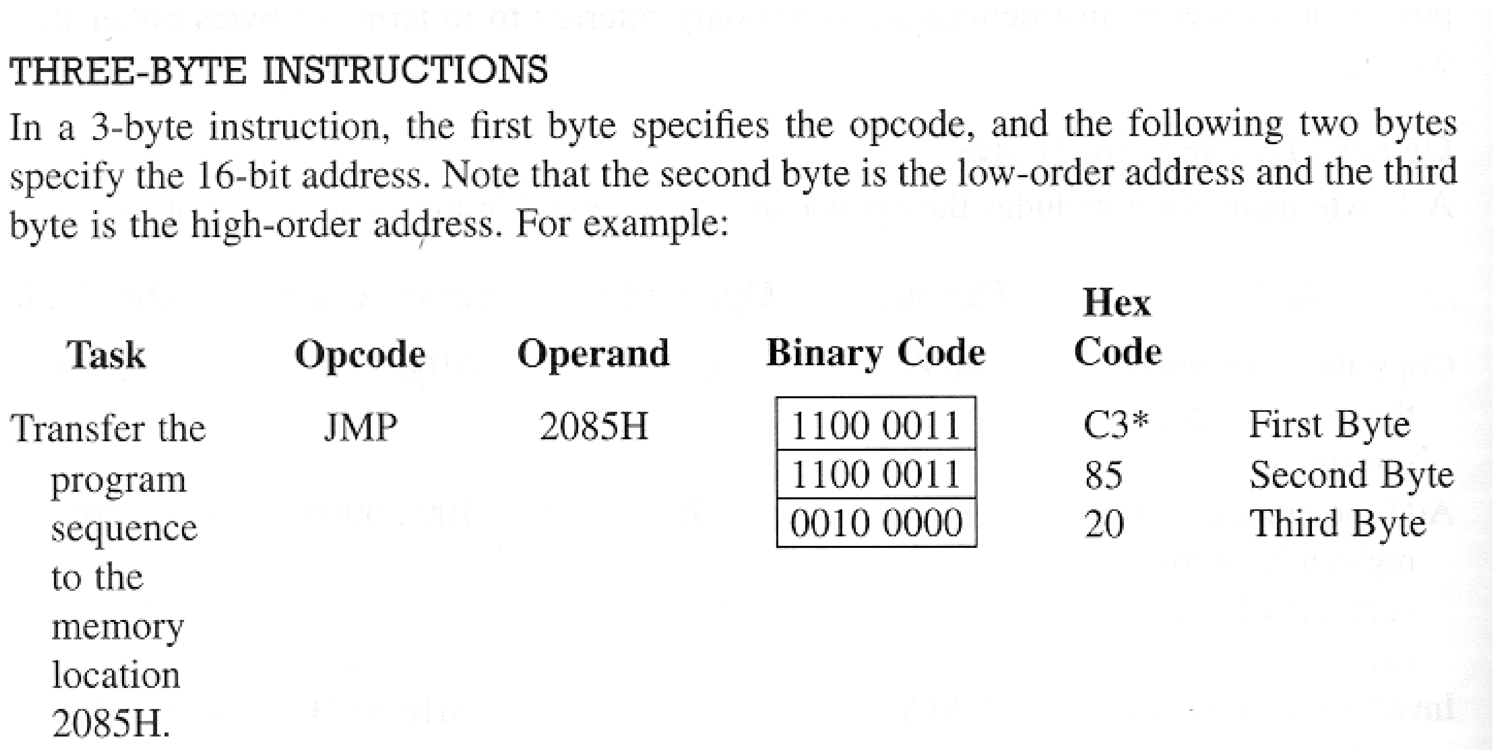
TWO-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A,Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

No FLAGS affected.

8085 Instruction Types



THREE-BYTE INSTRUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3*	First Byte
			1100 0011	85	Second Byte
			0010 0000	20	Third Byte

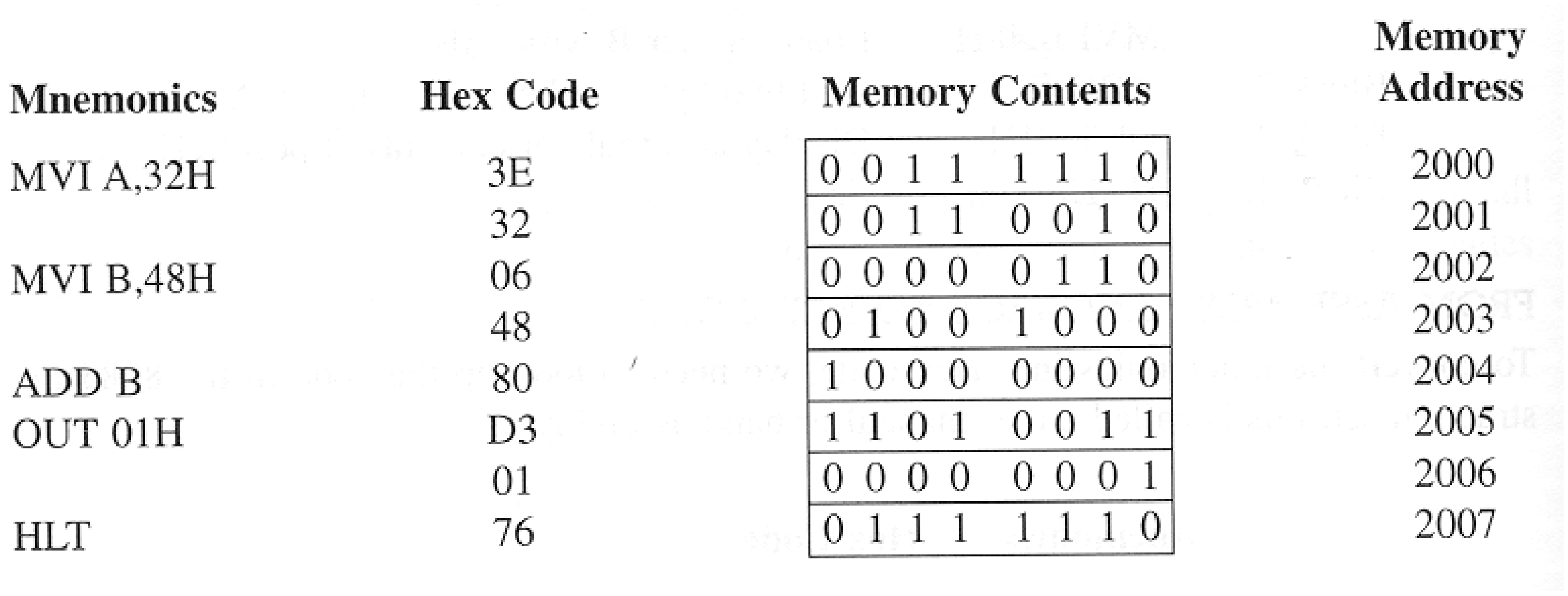
No FLAGS affected.

A VERY Simple Program

Add two hexadecimal numbers:

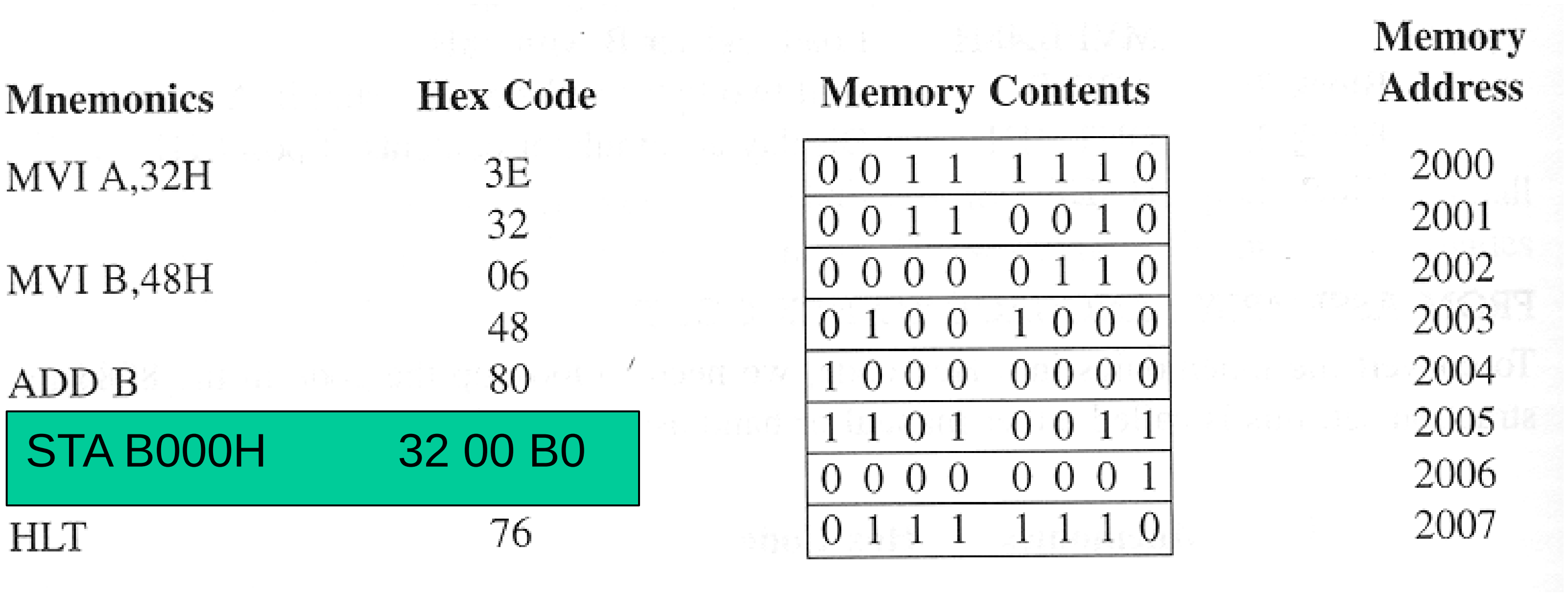
- Load register A (accumulator) with 32 (hex)
- Load register B with 48 (hex)
- Add the two numbers and save the sum in A
- Display accumulator (A) contents at port (01)
- End

A VERY Simple Program



Mnemonics	Hex Code	Memory Contents	Memory Address
MVI A,32H	3E	0 0 1 1 1 1 1 0	2000
	32	0 0 1 1 0 0 1 0	2001
MVI B,48H	06	0 0 0 0 0 1 1 0	2002
	48	0 1 0 0 1 0 0 0	2003
ADD B	80	1 0 0 0 0 0 0 0	2004
OUT 01H	D3	1 1 0 1 0 0 1 1	2005
	01	0 0 0 0 0 0 0 1	2006
HLT	76	0 1 1 1 1 1 1 0	2007

A VERY Simple Program – Version 2



Mnemonics	Hex Code	Memory Contents	Memory Address
MVI A,32H	3E	0 0 1 1 1 1 1 0	2000
	32	0 0 1 1 0 0 1 0	2001
MVI B,48H	06	0 0 0 0 0 1 1 0	2002
	48	0 1 0 0 1 0 0 0	2003
ADD B	80	1 0 0 0 0 0 0 0	2004
STA B000H	32 00 B0	1 1 0 1 0 0 1 1	2005
		0 0 0 0 0 0 0 1	2006
HLT	76	0 1 1 1 1 1 1 0	2007

Lab Problem – 1

- Assemble the following program. Also write what is the net effect of this program.

```
MVI  A, 8FH
MVI  B, 68H
SUB B
ANI   0FH
STA   2070H
HLT
```

Lab Problem – 2

- Assemble the following program assuming START is pointing to memory location B000H. Also write what is the net effect of this program.

```
START: LXI H, 2055H     ;index for data source
       LXI D, 2085H     ; index for data destination,
                        ; starting at last location
       MVI B, 06H       ; Byte Counter
NEXT:  MOV A, M         ; Get data byte
       STAX D           ; Store data byte
       INX H            ; Next location
       DCX D
       DCR B
       JNZ  NEXT        ; go back if counter is not 0
       HLT
```

Lab Problem – 3

- Assemble the following program assuming START is pointing to memory location c000H. Also write what is the net effect of this program.

```
START: LXI H, 2055H    ;index for data source
       LXI D, 2085H    ; index for data destination,
                       ; starting at last location
       MVI B, 06H      ; Byte Counter
NEXT:  MOV A, M        ; Get data byte
       STAX D          ; Store data byte
       INX H           ; Next location
       DCX D
       DCR B
       JNZ  NEXT       ; go back if counter is not 0
       HLT
```

Lab Problem – 4

- Write an 8085 assembly program to compute the sum of 10 numbers which are located in 10 consecutive byte positions starting from memory location 0xC000. The sum value has to be stored back into memory location 0xC00A. Assembly the program assuming that the program will be loaded started from the memory location 0xB000.

Detailed Review of the 8085 Instruction Set

Prof. Yusuf Leblebici
Microelectronic Systems Laboratory (LSM)

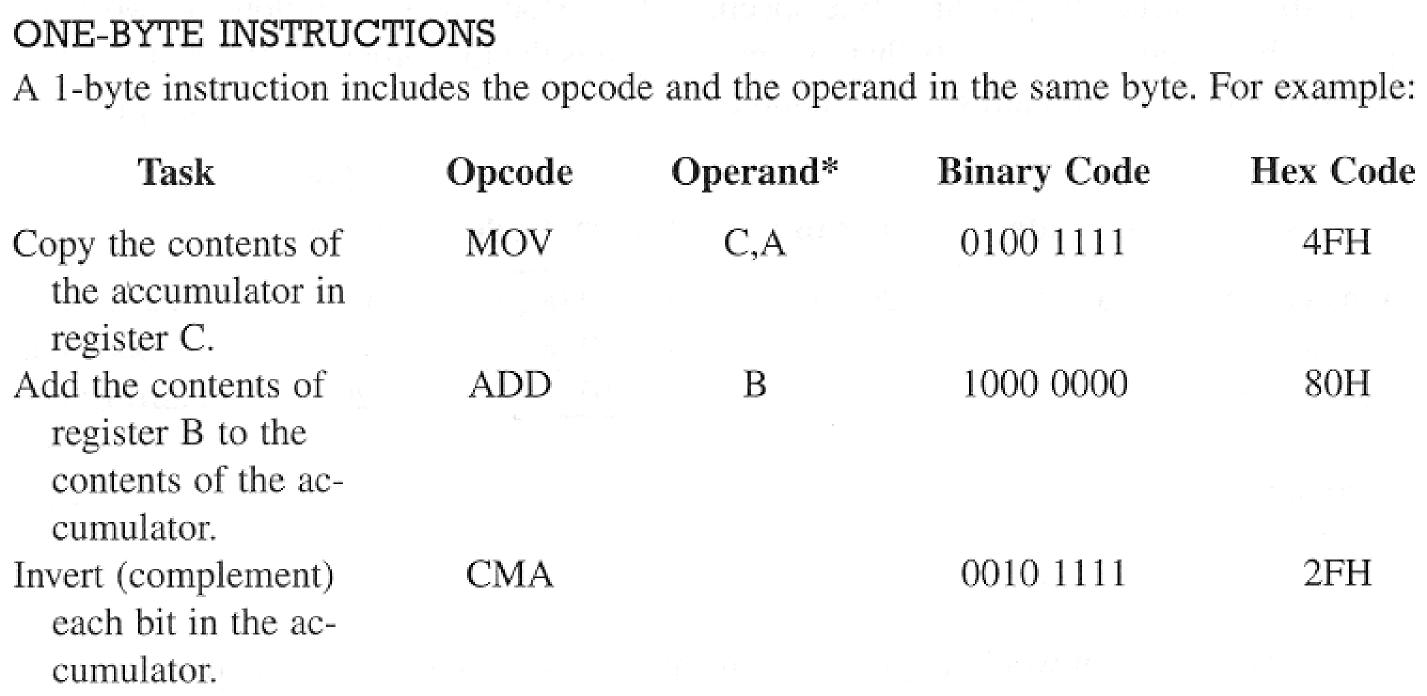
yusuf.leblebici@epfl.ch

8085 Instruction Set

The 8085 instructions can be classified as follows:

- Data transfer operations
  - Between registers
  - Between memory location and a register
  - Direct write to a register / memory
  - Between I/O device and accumulator
- Arithmetic operations (ADD, SUB, INR, DCR)
- Logic operations
- Branching operations (JMP, CALL, RET)

8085 Instruction Types



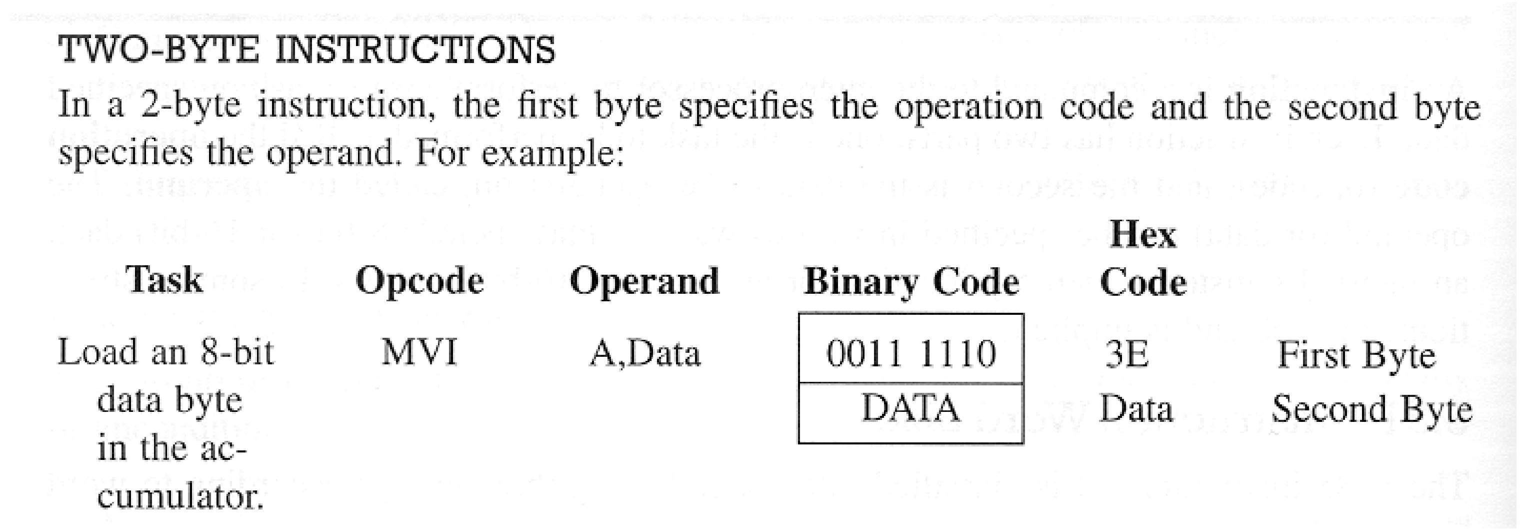
ONE-BYTE INSTRUCTIONS

A 1-byte instruction includes the opcode and the operand in the same byte. For example:

Task	Opcode	Operand*	Binary Code	Hex Code
Copy the contents of the accumulator in register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (complement) each bit in the accumulator.	CMA		0010 1111	2FH

1. MOV and CMA: No FLAGS affected.
2. ADD affects the FLAGS.

8085 Instruction Types

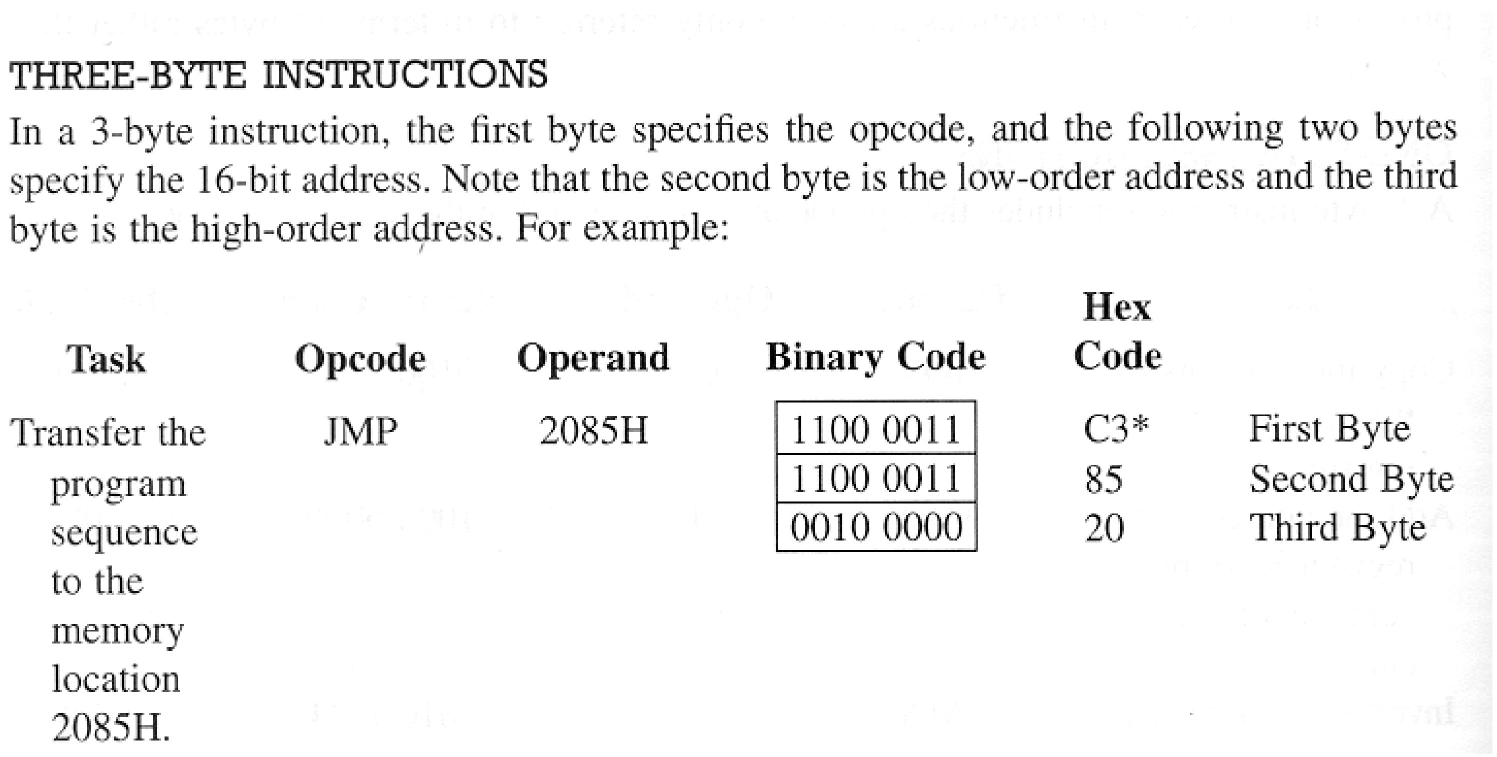


TWO-BYTE INSTRUCTIONS

In a 2-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the ac-cumulator.	MVI	A,Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

8085 Instruction Types

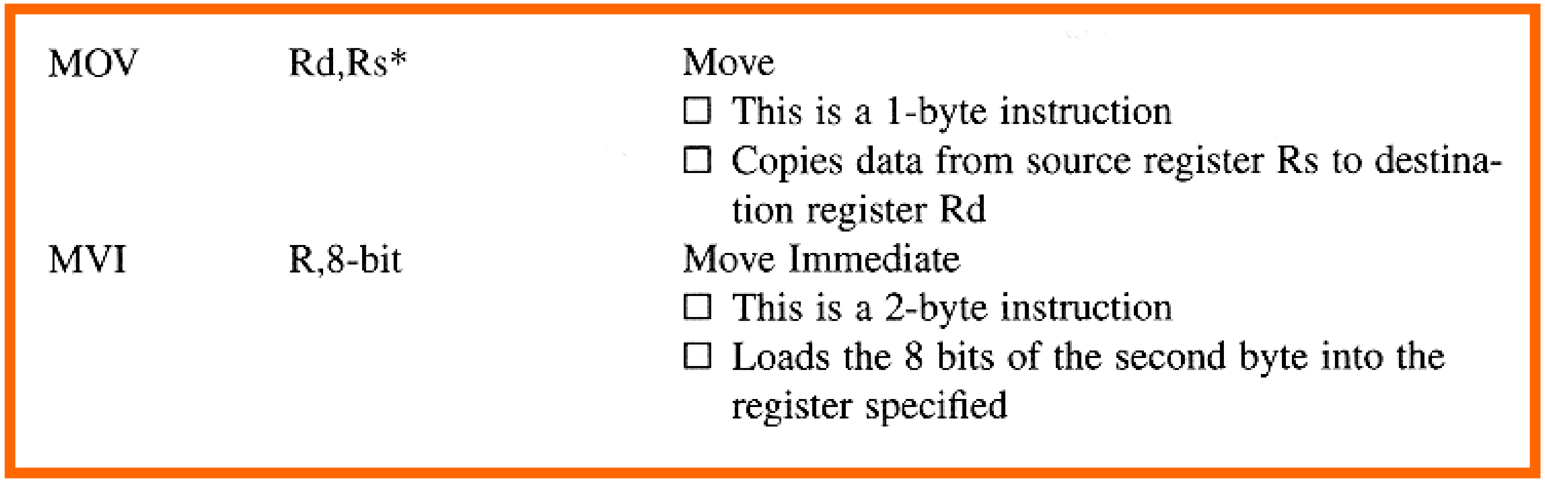


THREE-BYTE INSTRUCTIONS

In a 3-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011	C3*	First Byte
			1100 0011	85	Second Byte
			0010 0000	20	Third Byte

Simple Data Transfer Operations

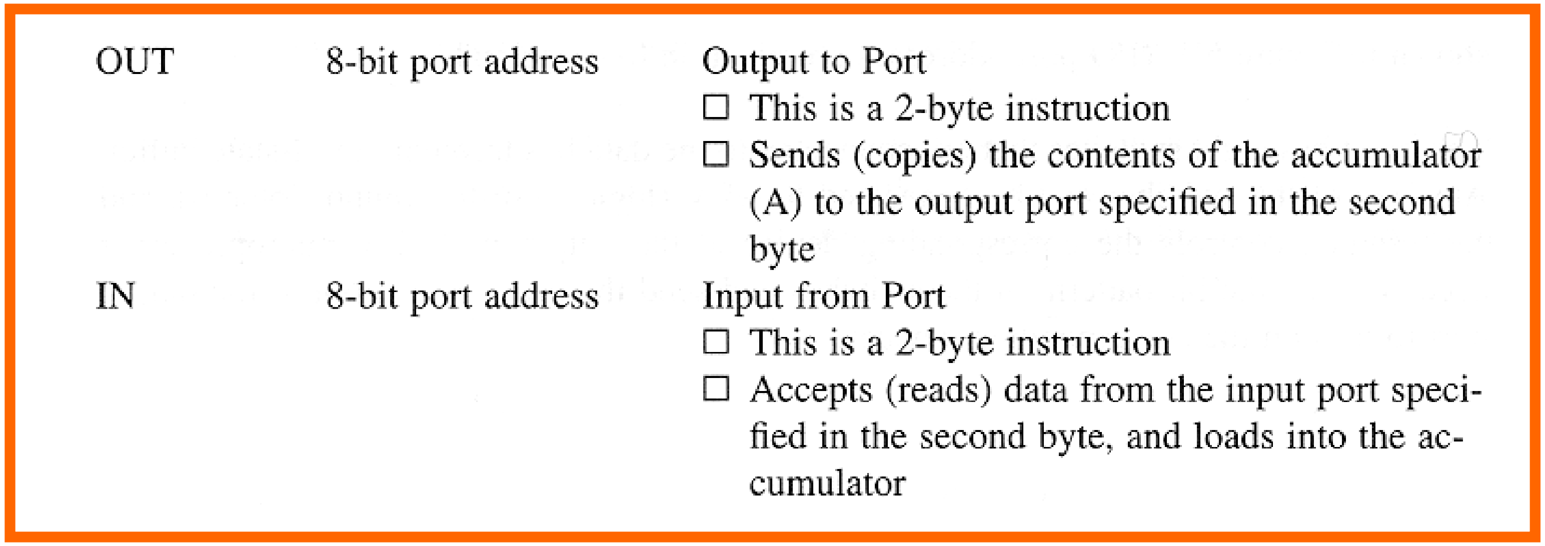


MOV	Rd,Rs*	Move ☐ This is a 1-byte instruction ☐ Copies data from source register Rs to destination register Rd
MVI	R,8-bit	Move Immediate ☐ This is a 2-byte instruction ☐ Loads the 8 bits of the second byte into the register specified

Examples:

MOV	B,A	47	From ACC to REG
MOV	C,D	4A	Between two REGs
MVI	D,47	16 47	Direct-write into REG D

Simple Data Transfer Operations



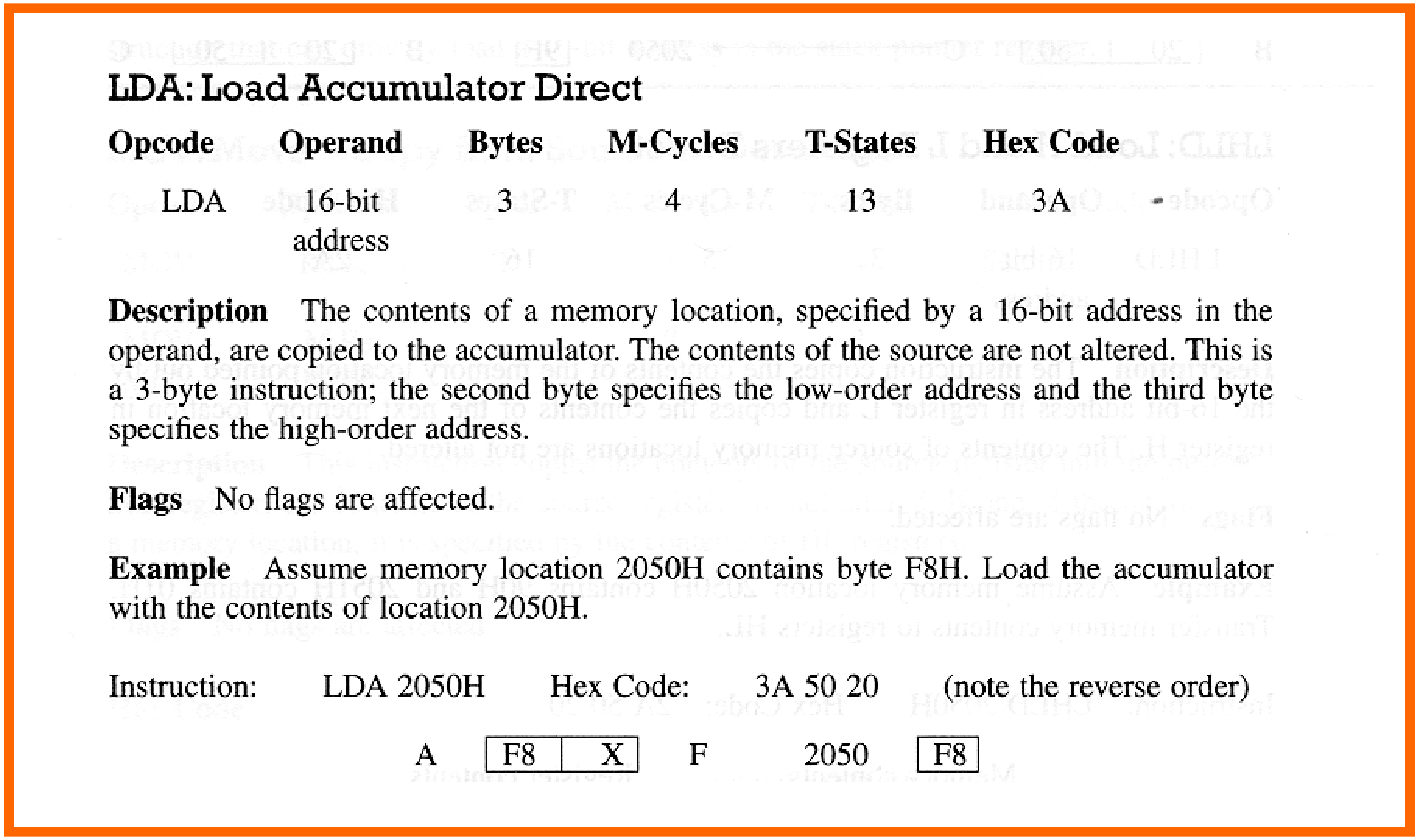
OUT	8-bit port address	Output to Port ☐ This is a 2-byte instruction ☐ Sends (copies) the contents of the accumulator (A) to the output port specified in the second byte
IN	8-bit port address	Input from Port ☐ This is a 2-byte instruction ☐ Accepts (reads) data from the input port specified in the second byte, and loads into the accumulator

Example:

- OUT 05 D3
  05

Contents of ACC sent to output port number 05.

Simple Memory Access Operations



LDA: Load Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDA	16-bit address	3	4	13	3A

Description The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

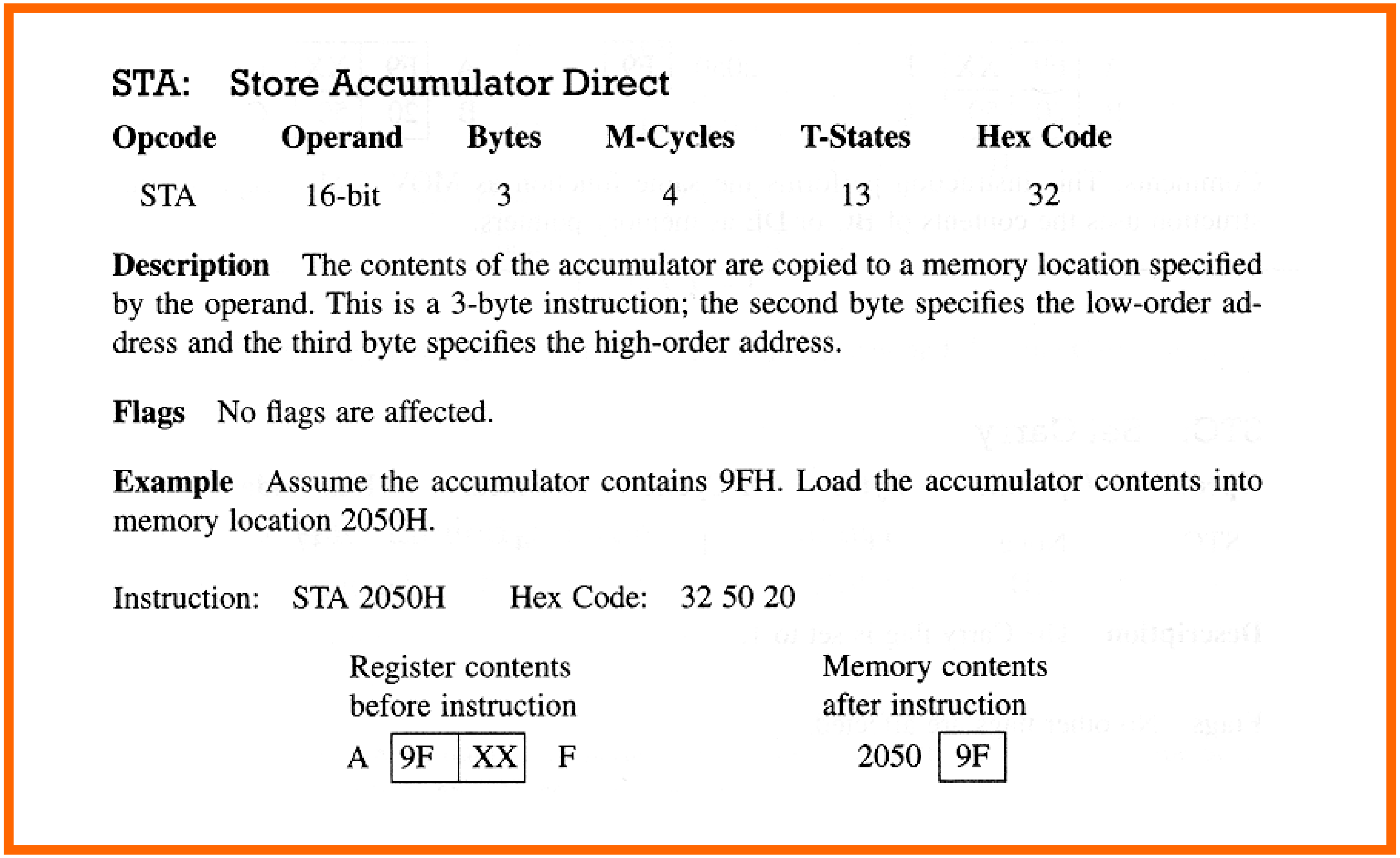
Flags No flags are affected.

Example Assume memory location 2050H contains byte F8H. Load the accumulator with the contents of location 2050H.

Instruction: LDA 2050H Hex Code: 3A 50 20 (note the reverse order)

A [F8 | X] F 2050 [F8]

Simple Memory Access Operations



STA: Store Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STA	16-bit	3	4	13	32

Description The contents of the accumulator are copied to a memory location specified by the operand. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the accumulator contains 9FH. Load the accumulator contents into memory location 2050H.

Instruction: STA 2050H Hex Code: 32 50 20

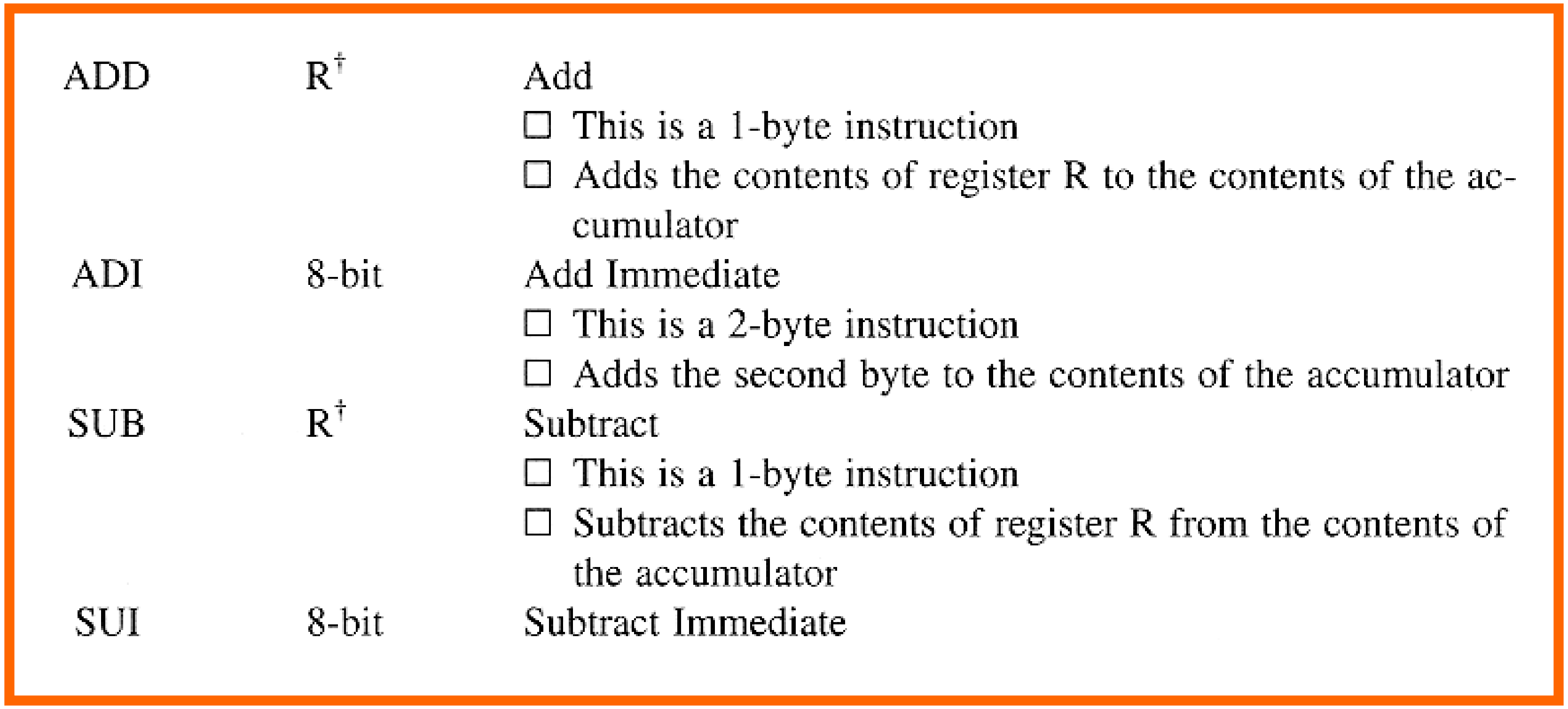
Register contents before instruction

A			F
A	9F	XX	F

Memory contents after instruction

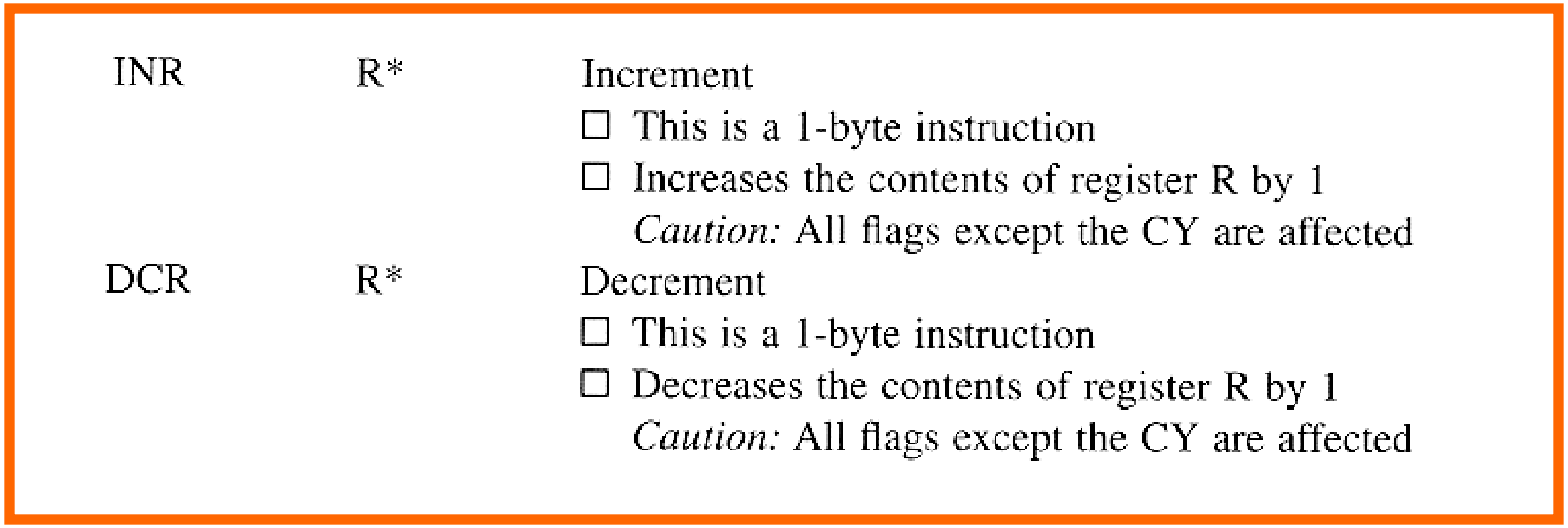
2050	9F

Arithmetic Operations



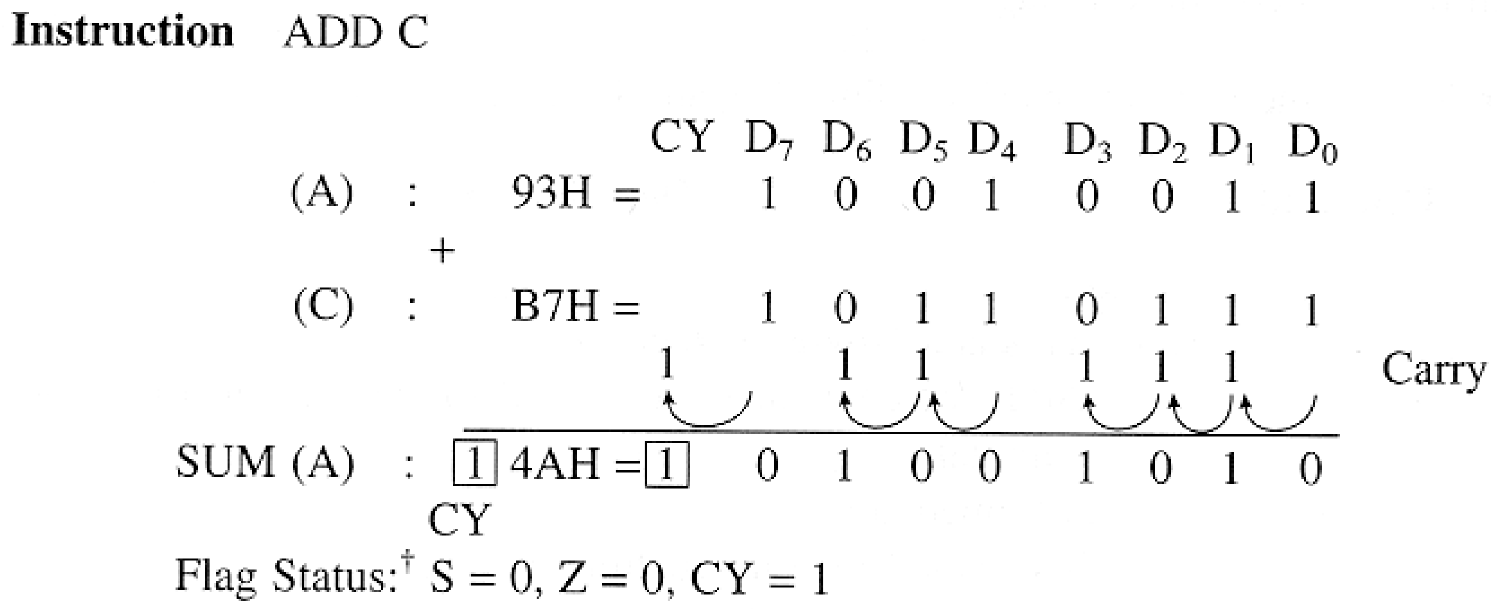
ADD	R[†]	Add ☐ This is a 1-byte instruction ☐ Adds the contents of register R to the contents of the ac-cumulator
ADI	8-bit	Add Immediate ☐ This is a 2-byte instruction ☐ Adds the second byte to the contents of the accumulator
SUB	R[†]	Subtract ☐ This is a 1-byte instruction ☐ Subtracts the contents of register R from the contents of the accumulator
SUI	8-bit	Subtract Immediate

Arithmetic Operations



INR	R*	Increment ☐ This is a 1-byte instruction ☐ Increases the contents of register R by 1 *Caution:* All flags except the CY are affected
DCR	R*	Decrement ☐ This is a 1-byte instruction ☐ Decreases the contents of register R by 1 *Caution:* All flags except the CY are affected

Arithmetic Operations

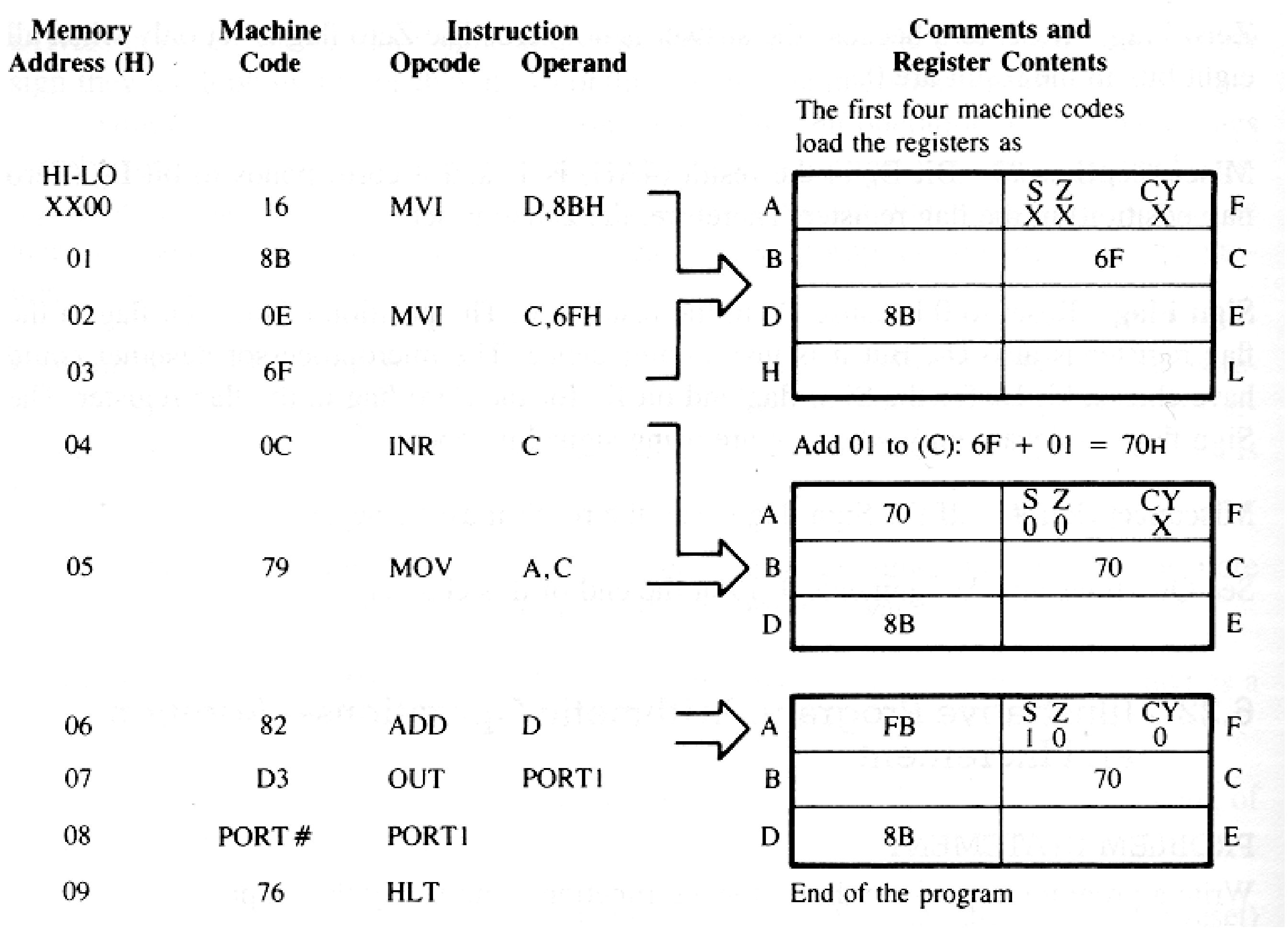


Instruction ADD C

			CY	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	
(A)	:	93H =		1	0	0	1	0	0	1	1	
	+											
(C)	:	B7H =		1	0	1	1	0	1	1	1	
			1		1	1		1	1	1		Carry
SUM (A)	:	$\boxed{1}$ 4AH =	$\boxed{1}$	0	1	0	0	1	0	1	0	
	CY											

Flag Status:[†] S = 0, Z = 0, CY = 1

Arithmetic Operations



Memory Address (H)	Machine Code	Instruction Opcode	Instruction Operand	Comments and Register Contents
HI-LO XX00	16	MVI	D,8BH	The first four machine codes load the registers as
01	8B			
02	0E	MVI	C,6FH	
03	6F			
04	0C	INR	C	Add 01 to (C): 6F + 01 = 70H
05	79	MOV	A,C	
06	82	ADD	D	
07	D3	OUT	PORT1	
08	PORT #	PORT1		
09	76	HLT		End of the program

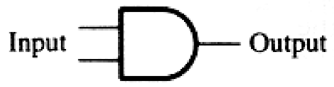
A		S Z X X CY X	F
B		6F	C
D	8B		E
H			L

A	70	S Z 0 0 CY X	F
B		70	C
D	8B		E

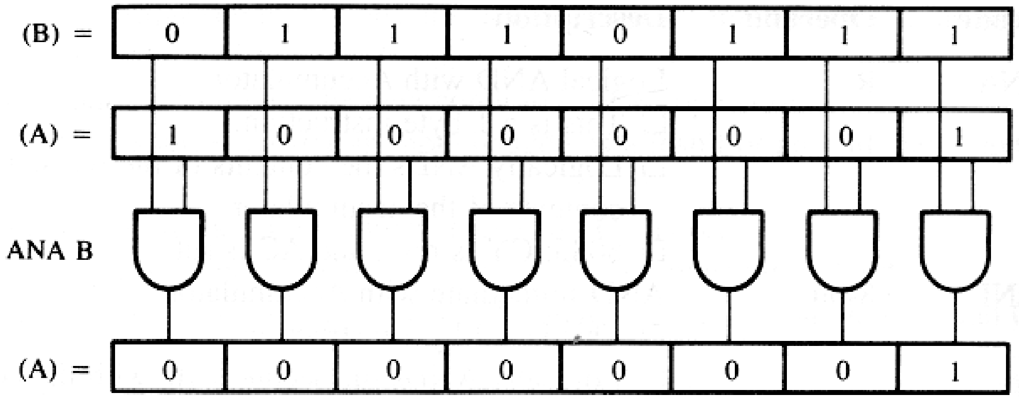
A	FB	S Z 1 0 CY 0	F
B		70	C
D	8B		E

Overview of Logic Operations

ANA:	AND	Logically AND the contents of a register.
ANI :	AND Immediate	Logically AND 8-bit data.
ORA:	OR	Logically OR the contents of a register.
ORI :	OR Immediate	Logically OR 8-bit data.
XRA:	X-OR	Exclusive-OR the contents of a register.
XRI :	X-OR Immediate	Exclusive-OR 8-bit data.

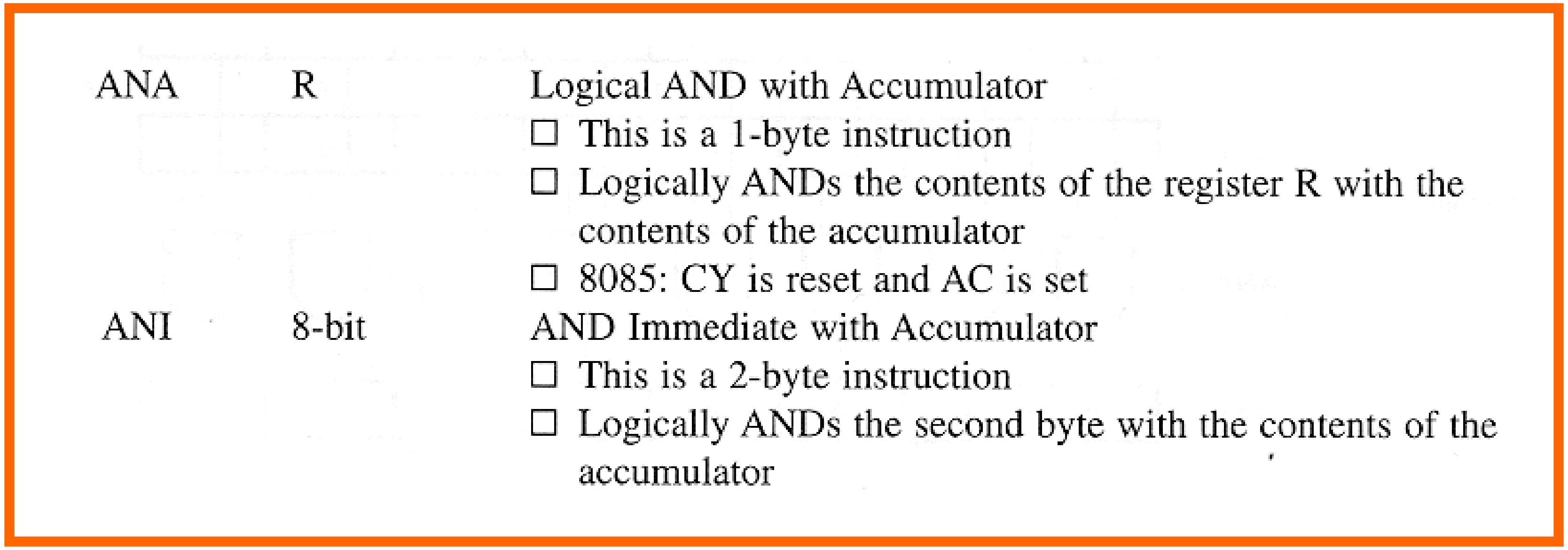


(a)



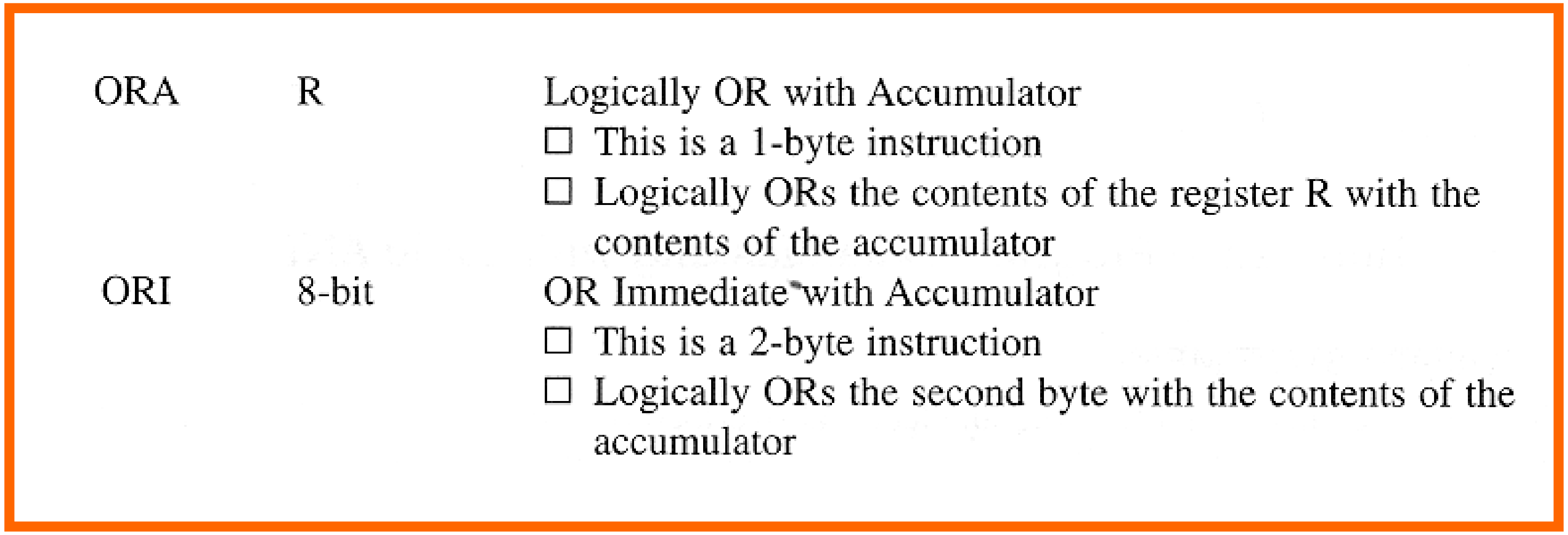
(b)

Logic Operations



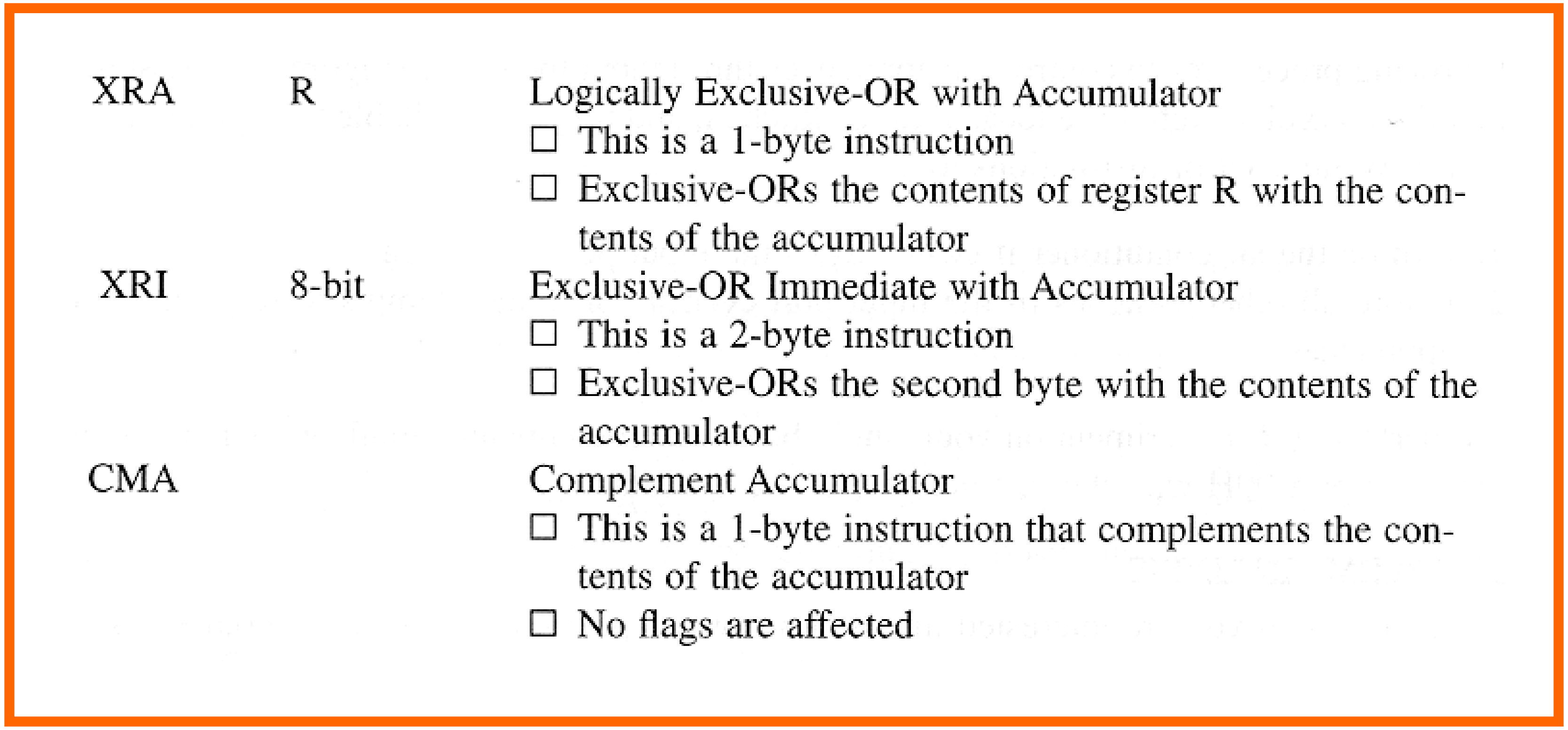
ANA	R	Logical AND with Accumulator ☐ This is a 1-byte instruction ☐ Logically ANDs the contents of the register R with the contents of the accumulator ☐ 8085: CY is reset and AC is set
ANI	8-bit	AND Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Logically ANDs the second byte with the contents of the accumulator

Logic Operations



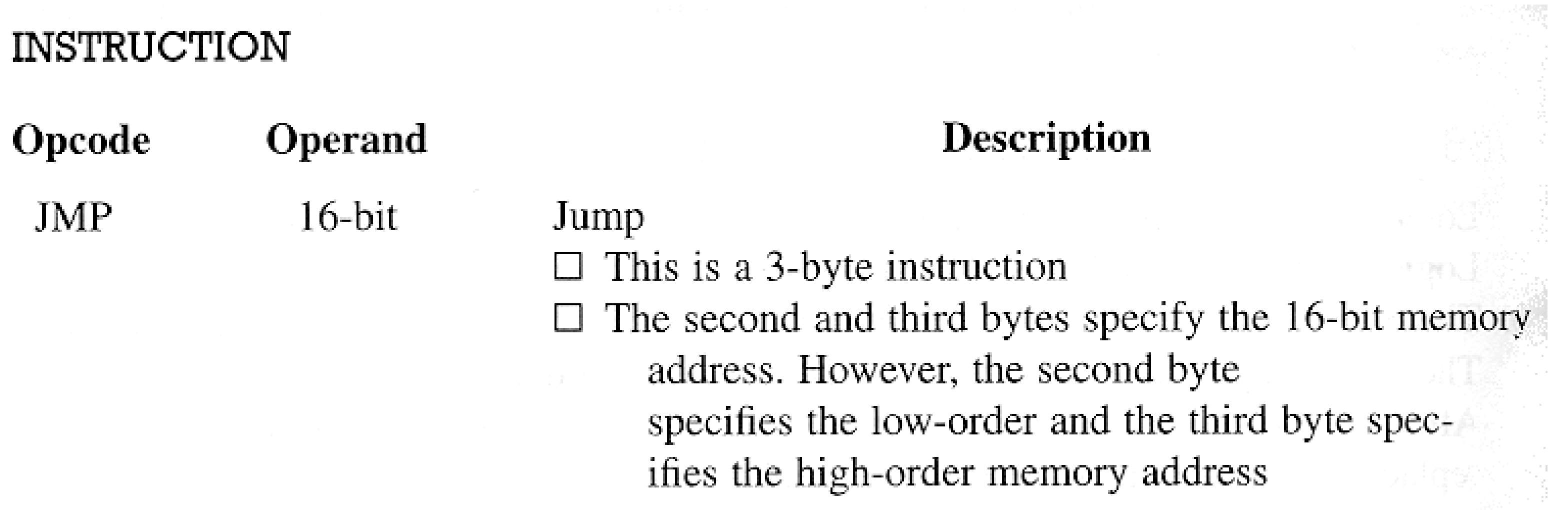
ORA	R	Logically OR with Accumulator ☐ This is a 1-byte instruction ☐ Logically ORs the contents of the register R with the contents of the accumulator
ORI	8-bit	OR Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Logically ORs the second byte with the contents of the accumulator

Logic Operations



XRA	R	Logically Exclusive-OR with Accumulator ☐ This is a 1-byte instruction ☐ Exclusive-ORs the contents of register R with the contents of the accumulator
XRI	8-bit	Exclusive-OR Immediate with Accumulator ☐ This is a 2-byte instruction ☐ Exclusive-ORs the second byte with the contents of the accumulator
CMA		Complement Accumulator ☐ This is a 1-byte instruction that complements the contents of the accumulator ☐ No flags are affected

Branching Operations

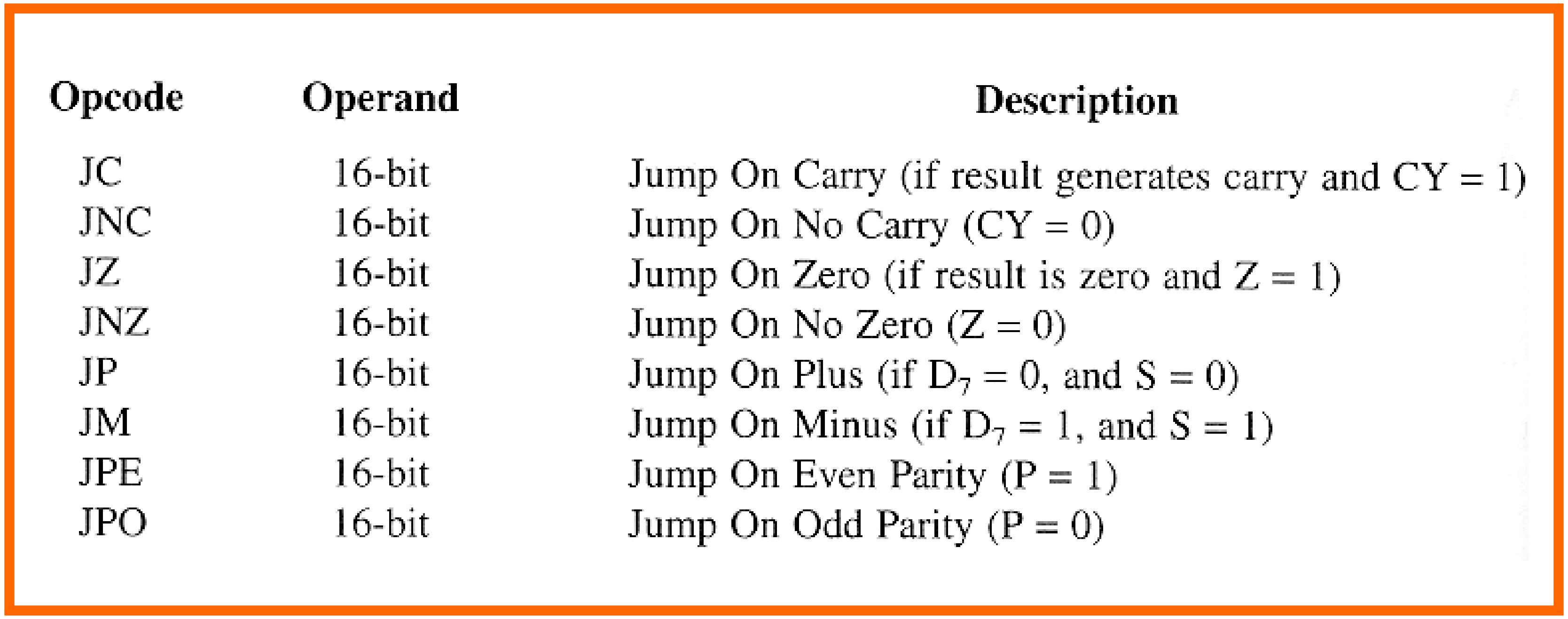


INSTRUCTION

Opcode	Operand	Description
JMP	16-bit	Jump ☐ This is a 3-byte instruction ☐ The second and third bytes specify the 16-bit memory address. However, the second byte specifies the low-order and the third byte specifies the high-order memory address

Note: This is an **unconditional** jump operation.
It will **always** force the program counter to a fixed memory address $\longrightarrow$ continuous loop !

Branching Operations



Opcode	Operand	Description
JC	16-bit	Jump On Carry (if result generates carry and CY = 1)
JNC	16-bit	Jump On No Carry (CY = 0)
JZ	16-bit	Jump On Zero (if result is zero and Z = 1)
JNZ	16-bit	Jump On No Zero (Z = 0)
JP	16-bit	Jump On Plus (if $D_7 = 0$, and S = 0)
JM	16-bit	Jump On Minus (if $D_7 = 1$, and S = 1)
JPE	16-bit	Jump On Even Parity (P = 1)
JPO	16-bit	Jump On Odd Parity (P = 0)

Conditional jump operations are very useful for decision making during the execution of the program.

Example

Write a 8085 machine code program:

- Read two different memory locations
- Add the contents
- Send the result to output port 02 (display) if there is no overflow
- Display “FF” if there is an overflow
- Stop

Example

2000	LDA	2050	3A	Load contents of memory location 2050 into accumulator
2001			50	
2002			20	
2003	MOV	B,A	47	Save the first number in B
2004	LDA	2051	3A	Load contents of memory location 2051 into accumulator
2005			51	
2006			20	
2007	ADD	B	80	Add accumulator with B
2008	JNC	XXYY	D2	Jump to **YYXX** if no carry !
2009			YY	
2010			XX	
2011	MVI	A,FF	3E	Direct write FF into accumulator
2012			FF	
2013	OUT	02	D3	Display accumulator contents at output port 02
2014			02	
2015	HLT		76	Stop

Updated Code

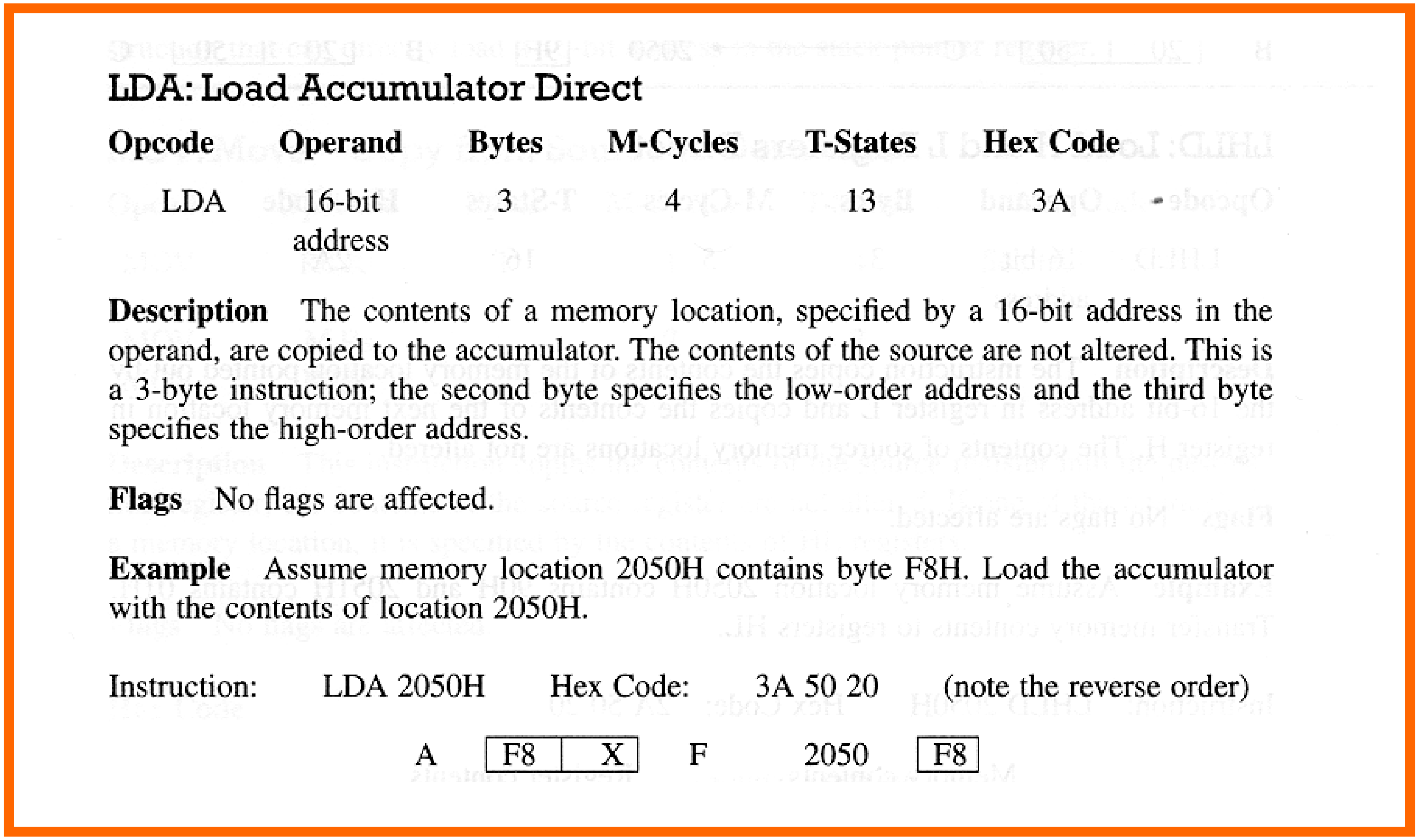
2000	LDA	2050	3A	Load contents of memory location 2050 into accumulator
2001			50	
2002			20	
2003	MOV	B,A	47	Save the first number in B
2004	LDA	2051	3A	Load contents of memory location 2051 into accumulator
2005			51	
2006			20	
2007	ADD	B	80	Add accumulator with B
2008	JNC	2013	D2	Jump to **2013** if no carry !
2009			13	
2010			20	
2011	MVI	A,FF	3E	Direct write FF into accumulator
2012			FF	
2013	OUT	02	D3	Display accumulator contents at output port 02
2014			02	
2015	HLT		76	Stop

Indirect Memory Access

Prof. Yusuf Leblebici
Microelectronic Systems Laboratory (LSM)

yusuf.leblebici@epfl.ch

Direct Memory Access Operations



LDA: Load Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
LDA	16-bit address	3	4	13	3A

Description The contents of a memory location, specified by a 16-bit address in the operand, are copied to the accumulator. The contents of the source are not altered. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

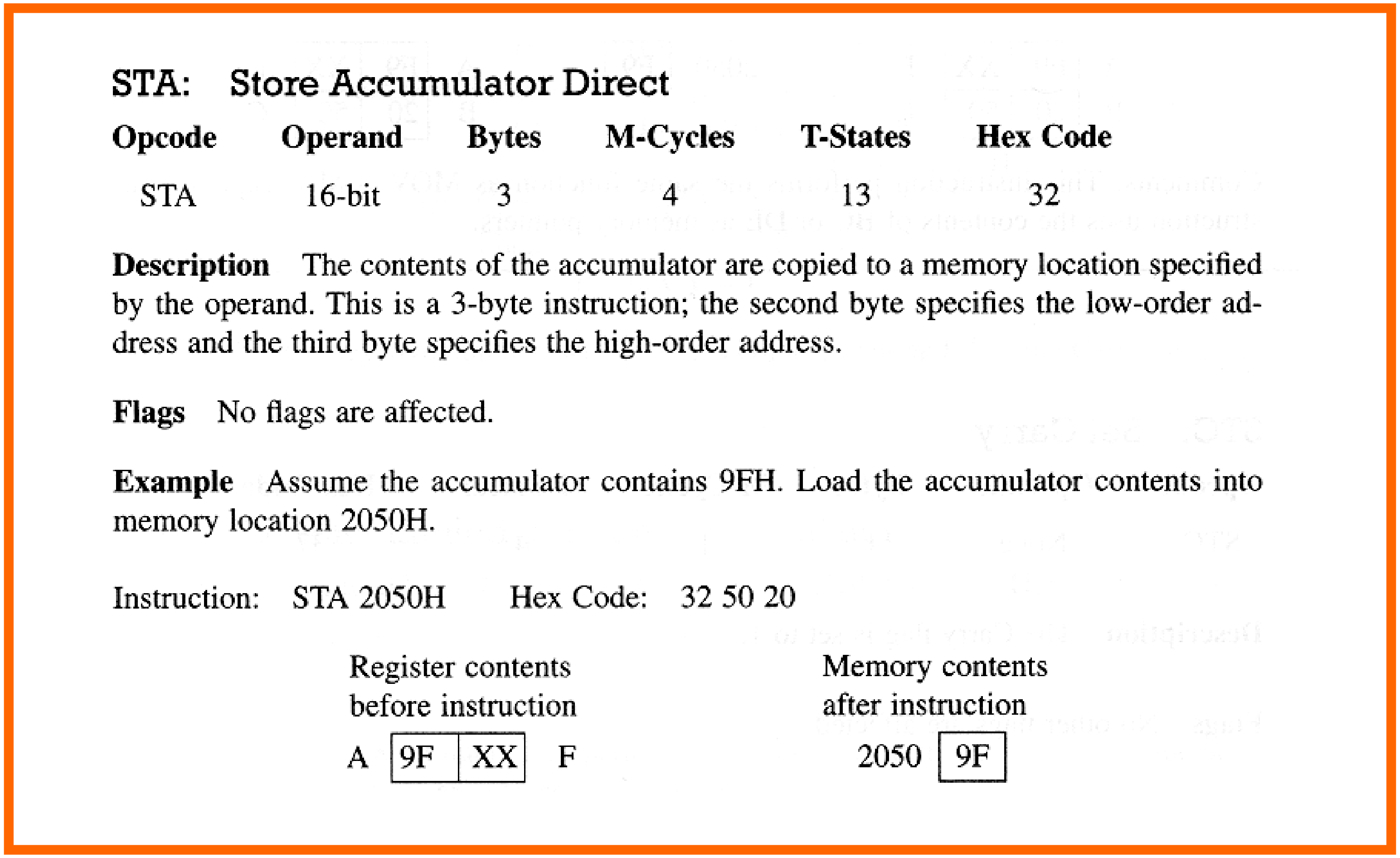
Flags No flags are affected.

Example Assume memory location 2050H contains byte F8H. Load the accumulator with the contents of location 2050H.

Instruction: LDA 2050H Hex Code: 3A 50 20 (note the reverse order)

A | F8 | X | F 2050 | F8 |

Direct Memory Access Operations



STA: Store Accumulator Direct

Opcode	Operand	Bytes	M-Cycles	T-States	Hex Code
STA	16-bit	3	4	13	32

Description The contents of the accumulator are copied to a memory location specified by the operand. This is a 3-byte instruction; the second byte specifies the low-order address and the third byte specifies the high-order address.

Flags No flags are affected.

Example Assume the accumulator contains 9FH. Load the accumulator contents into memory location 2050H.

Instruction: STA 2050H Hex Code: 32 50 20

Register contents before instruction

A	9F	XX	F

Memory contents after instruction

2050	9F

Example

Write a 8085 machine code program:

- Read two different memory locations
- Add the contents
- Send the result to output port 02 (display) if there is no overflow
- Display “FF” if there is an overflow
- Stop

Example

2000	LDA	2050	3A	Load contents of memory location 2050 into accumulator
2001			50	
2002			20	
2003	MOV	B,A	47	Save the first number in B
2004	LDA	2051	3A	Load contents of memory location 2051 into accumulator
2005			51	
2006			20	
2007	ADD	B	80	Add accumulator with B
2008	JNC	XXYY	D2	Jump to **YYXX** if no carry !
2009			YY	
2010			XX	
2011	MVI	A,FF	3E	Direct write FF into accumulator
2012			FF	
2013	OUT	02	D3	Display accumulator contents at output port 02
2014			02	
2015	HLT		76	Stop

Updated Code

2000	LDA	2050	3A	Load contents of memory location 2050 into accumulator
2001			50	
2002			20	
2003	MOV	B,A	47	Save the first number in B
2004	LDA	2051	3A	Load contents of memory location 2051 into accumulator
2005			51	
2006			20	
2007	ADD	B	80	Add accumulator with B
2008	JNC	2013	D2	Jump to **2013** if no carry !
2009			13	
2010			20	
2011	MVI	A,FF	3E	Direct write FF into accumulator
2012			FF	
2013	OUT	02	D3	Display accumulator contents at output port 02
2014			02	
2015	HLT		76	Stop

Indirect Memory Access Operations

- Use a register PAIR as an address pointer !

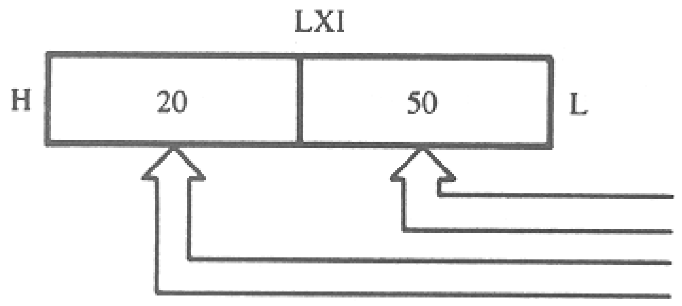
- We can define memory access operations using the memory location (16 bit address) stored in a register pair: BC, DE or HL.

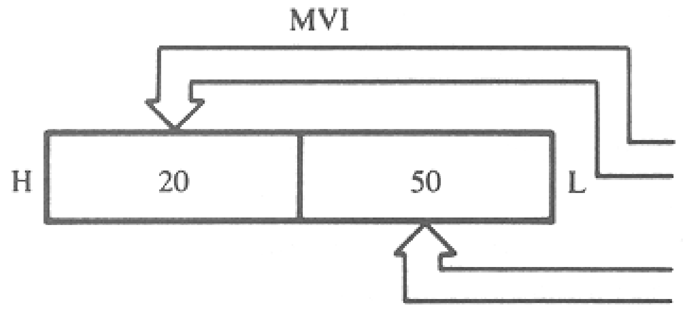
- First, we have be able to **load** the register pairs.

```
LXI B, (16-bit address)
LXI D, (16-bit address)
LXI H, (16-bit address)
```

- We can also increment / decrement register pairs.

Loading Register Pairs





Machine Code	Mnemonics	Comments
21	LXI H,2050H	;Load HL registers
50*		;50H in L register and
20		;20H in H register
26	MVI H,20H	;Load 20H in register H
20		
2E	MVI L,50H	;Load 50H in register L
50		

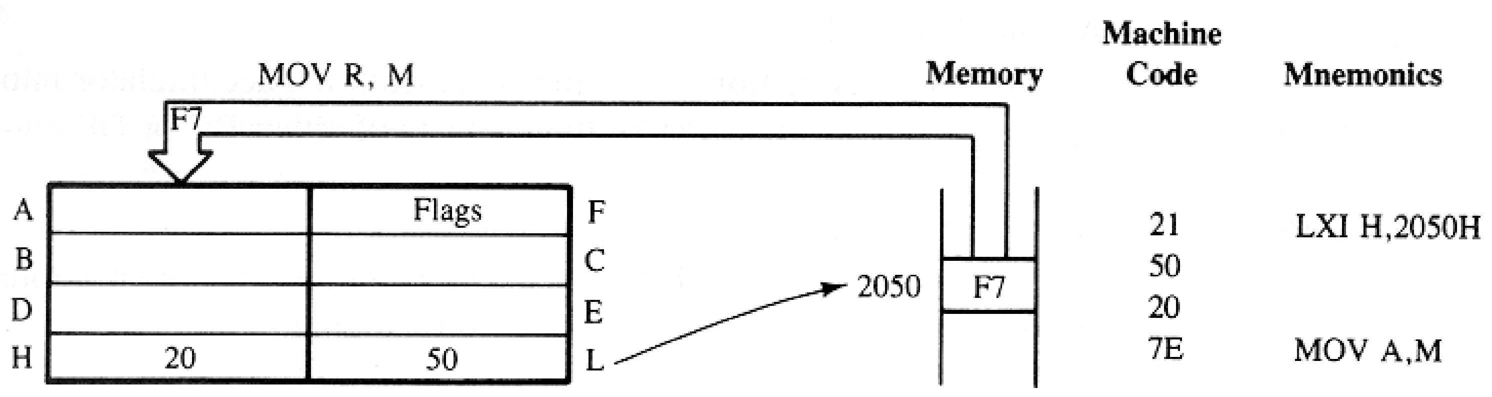
Data Transfer from Memory to Processor

- Once the register pairs are loaded with the memory address, we can use them as **pointers**.

- MOV R,M
  Move the contents of the memory location stored in HL register pair into register (R).

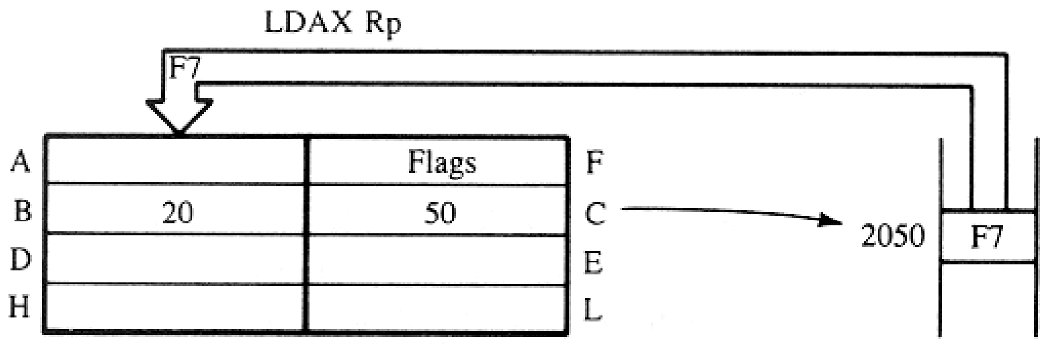
- LDAX B / LDAX D
  Move the contents of the memory location stored in BC (or DE) register pair into the accumulator.

Data Transfer from Memory to Processor



Machine Code	Mnemonics
21	LXI H,2050H
50	
20	
7E	MOV A,M

(a)



01	LXI B,2050H
50	
20	
0A	LDAX B

(b)

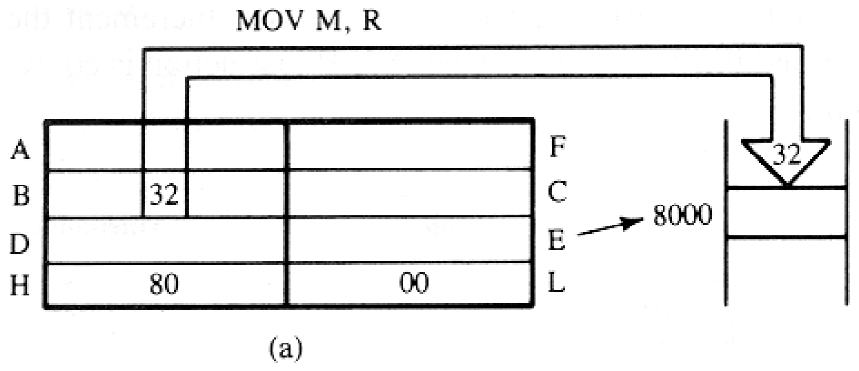
Data Transfer from Processor to Memory

- Once the register pairs are loaded with the memory address, we can use them as **pointers**.

- MOV M,R
  Move the contents of the register (R) into memory location stored in HL register pair.

- STAX B / STAX D
  Move the contents of the accumulator into the memory location stored in BC (or DE) register pair.

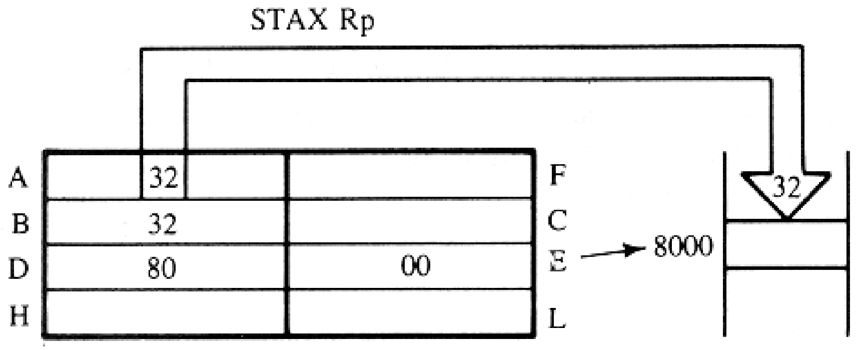
Data Transfer from Processor to Memory



(a)

Machine Code	Mnemonics
21	LXI H,8000H
00	
80	
70	MOV M,B

This instruction copies the contents of the accumulator into memory. Therefore, it is necessary first to copy (B) into A.



(b)

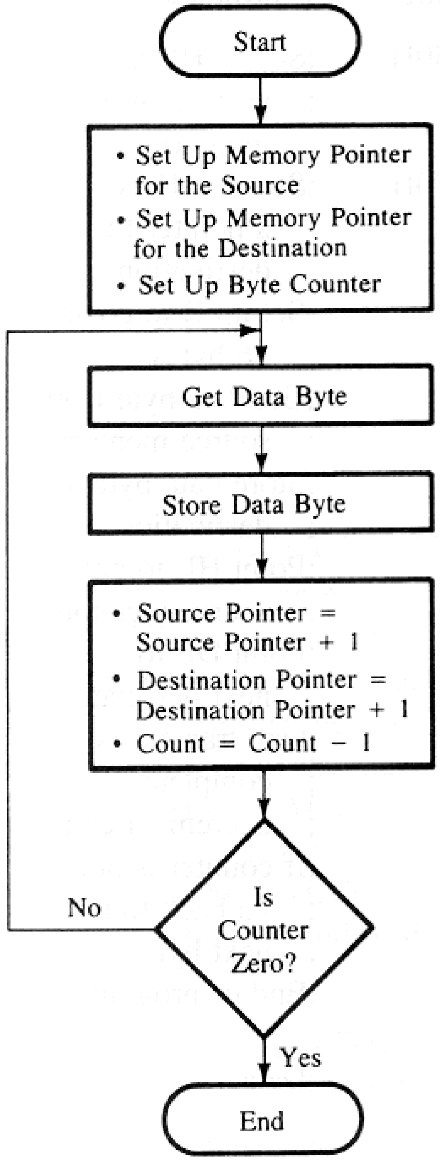
11	LXI D,8000H
00	
80	
78	MOV A,B
12	STAX D

Block Data Transfer Example

- 16 bytes of data are stored in memory locations XX50 to XX5F (16 consecutive memory addresses).

- Transfer (copy) the entire block of data to new memory locations starting at XX70.

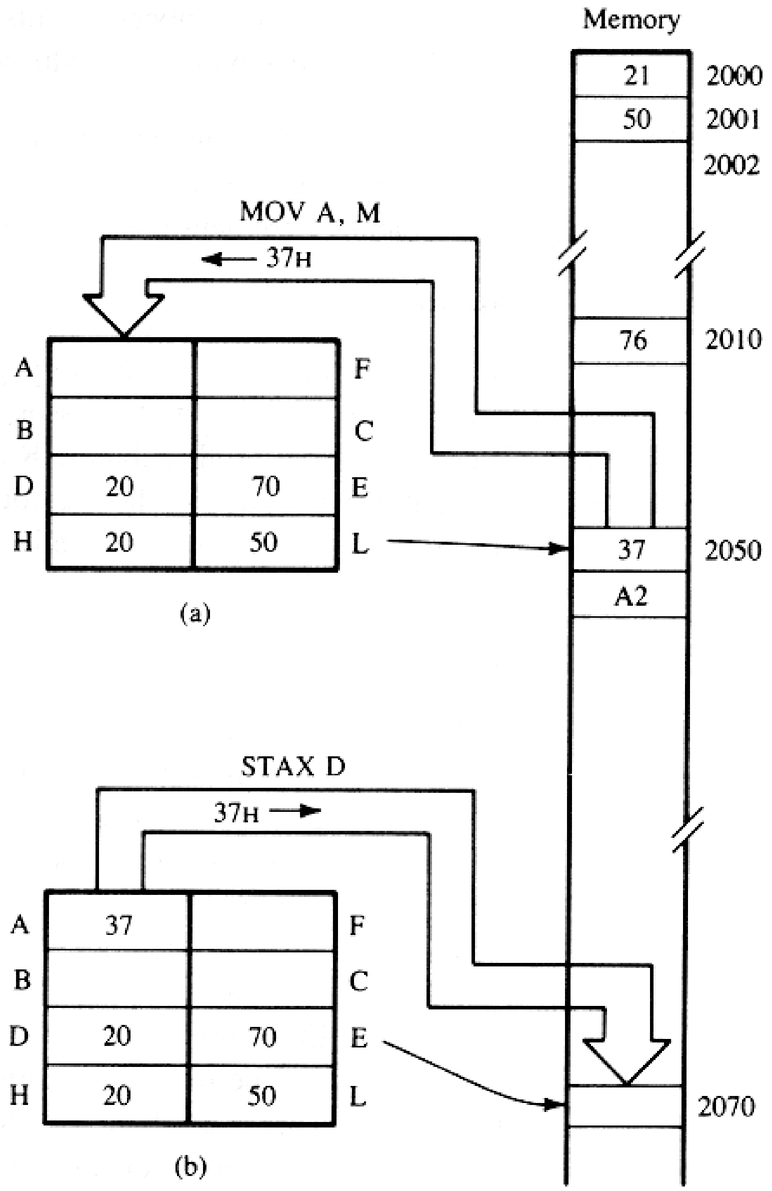
- Make use of memory address pointers and indirect addressing !


Start
• Set Up Memory Pointer for the Source
• Set Up Memory Pointer for the Destination
• Set Up Byte Counter
Get Data Byte
Store Data Byte
• Source Pointer = Source Pointer + 1
• Destination Pointer = Destination Pointer + 1
• Count = Count − 1
Is Counter Zero?
No
Yes
End

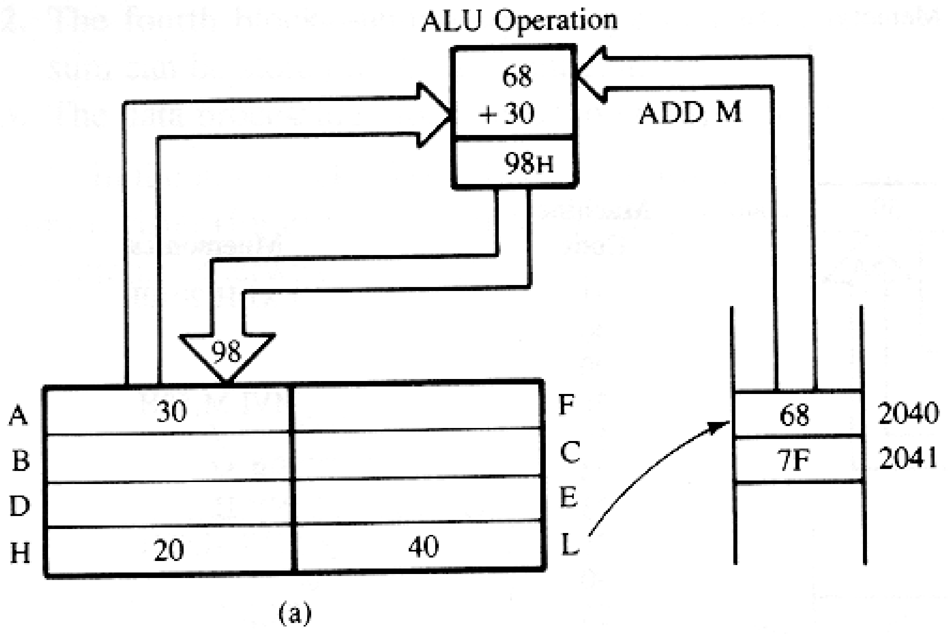
	Mnemonics	Steps
Block 1	LXI H,XX50H	**1.** Set up HL as a pointer for the source memory
	LXI D,XX70H	Set up DE as a pointer for the destination memory
	MVI B,10H	Set up B as byte counter
Block 2	NEXT: MOV A,M	**2.** Get data byte from the source memory
Block 7	STAX D	**3.** Store the data byte in destination memory
Block 5	INX H INX D DCR B	**4.** Get ready to transfer next byte
Block 6	JNZ NEXT	**5.** Go back to get next byte if byte counter ≠0
	HLT	

PROGRAM

Memory Address	Hex		Instructions		
HI-LO	Code	Label	Opcode	Operand	Comments
XX00	21	START:	LXI	H,XX50H	;Set up HL as a
01	50				; pointer for source
02	XX				; memory
03	11		LXI	D,XX70H	;Set up DE as
04	70				; a pointer for
05	XX				; destination
06	06		MVI	B,10H	;Set up B to count
07	10				; 16 bytes
08	7E	NEXT:	MOV	A,M	;Get data byte from
					; source memory
09	12		STAX	D	;Store data byte at
					; destination
0A	23		INX	H	;Point HL to next
					; source location
0B	13		INX	D	;Point DE to
					; next destination
0C	05		DCR	B	;One transfer is
					; complete,
					; decrement count
0D	C2		JNZ	NEXT	;If counter is not 0,
0E	08				; go back to transfer
0F	XX				; next byte
10	76		HLT		;End of program
XX50	37				;Data
↓	↓				
XX5F	98				


Memory
21
2000
50
2001
2002
MOV A, M
37H
76
2010
A
F
B
C
D
20
70
E
H
20
50
L
37
2050
A2
(a)
STAX D
37H
A
37
F
B
C
D
20
70
E
H
20
50
L
2070
(b)

Arithmetic Operations Using Memory



(a)

Machine Code	Mnemonics
21	LXI H,2040H
40	
20	
86	ADD M