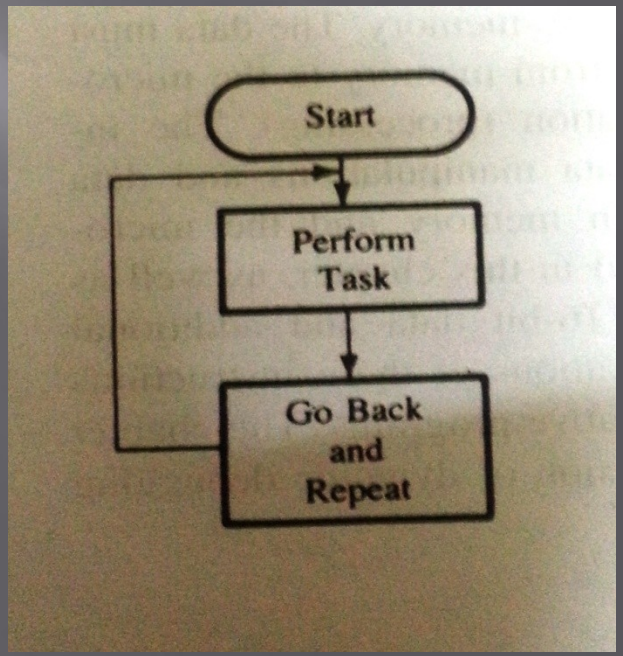
Programming Techniques with Additional Instructions

Lecture 1

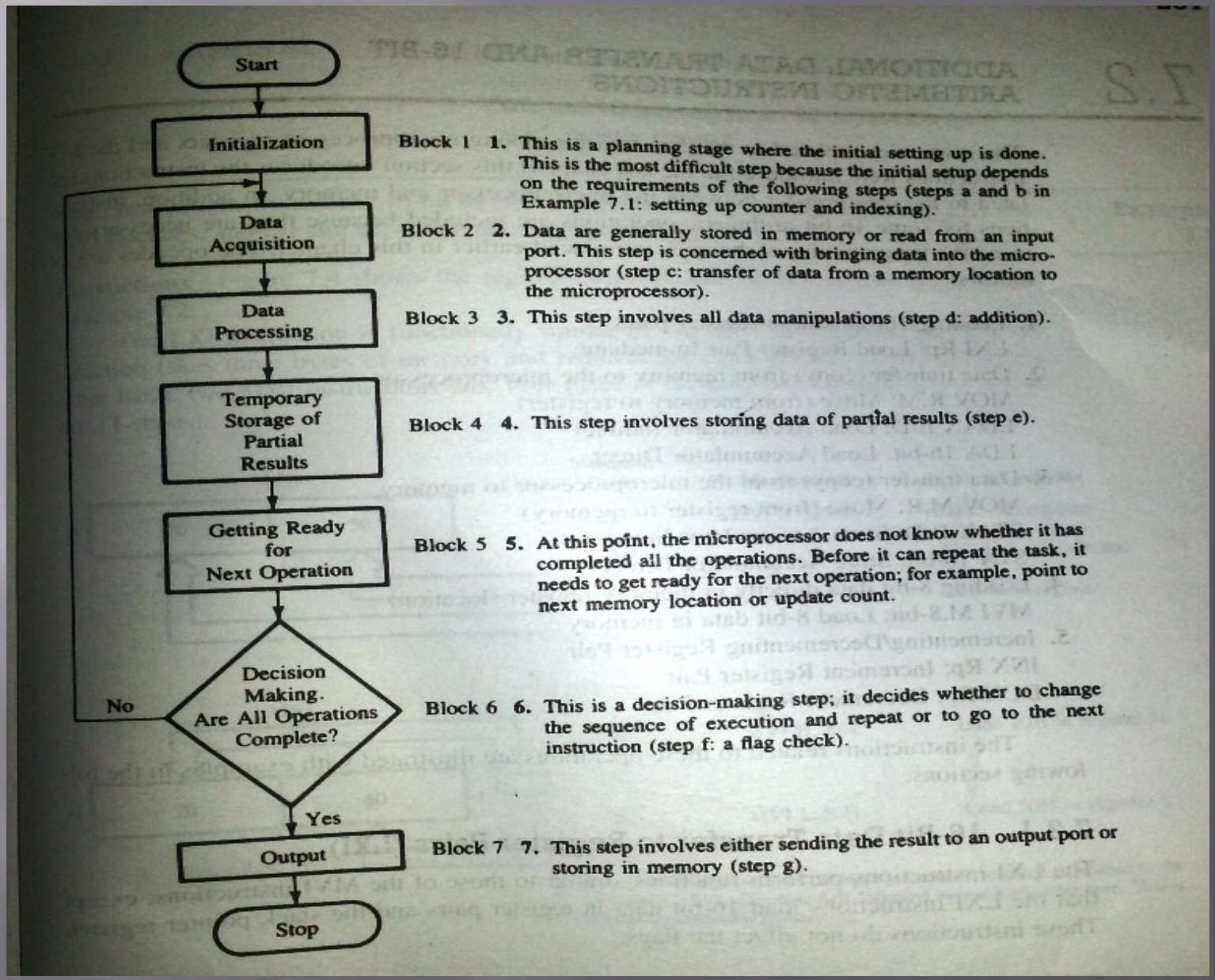
Looping, counting and Indexing

- Continuous loop-repeat task continuously
- Conditional loop-repeats a task until certain data conditions are met.

Flowchart of continuous loop



Generalized programming Flowchart



Block 1 **1.** This is a planning stage where the initial setting up is done. This is the most difficult step because the initial setup depends on the requirements of the following steps (steps a and b in Example 7.1: setting up counter and indexing).

Block 2 **2.** Data are generally stored in memory or read from an input port. This step is concerned with bringing data into the microprocessor (step c: transfer of data from a memory location to the microprocessor).

Block 3 **3.** This step involves all data manipulations (step d: addition).

Block 4 **4.** This step involves storing data of partial results (step e).

Block 5 **5.** At this point, the microprocessor does not know whether it has completed all the operations. Before it can repeat the task, it needs to get ready for the next operation; for example, point to next memory location or update count.

Block 6 **6.** This is a decision-making step; it decides whether to change the sequence of execution and repeat or to go to the next instruction (step f: a flag check).

Block 7 **7.** This step involves either sending the result to an output port or storing in memory (step g).

16 BIT DATA TRANSFER TO REGISTER PAIR (LXI)

Load register pair immediate

- LXI Reg. pair, 16-bit data.
- The instruction loads 16-bit data in the register pair designated in the operand.
- Example: LXI H, 2034H or LXI H, XYZ

Load H and L registers direct

- LHLD 16-bit address
- The instruction copies the contents of the memory location pointed out by the 16-bit address into register L and copies the contents of the next memory location into register H.
- The contents of source memory locations are not altered.

Example: LHLD 2040H

DATA TRANSFER FROM MEMORY TO MICROPROCESSOR

MOV R,M

- R, M copies data byte from Memory to Register. Memory location, its location is specified by the contents of the HL registers.
- Example: MOV B, M
- Load accumulator indirect

LDAX B/D Reg. pair

- The contents of the designated register pair point to a memory
- location. This instruction copies the contents of that memory
- location into the accumulator. The contents of either the
- register pair or the memory location are not altered.
- Example: LDAX B

Load accumulator

- LDA 16-bit address 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.
- Example: LDA 2034H

DATA TRANSFER FROM MICROPROCESSOR TO MEMORY OR DIRECTLY INTO MEMORY

MOV M,R

- This instruction copies the contents of the source.
- The source register are not altered. As one of the operands is a memory location, its location is specified by the contents of the HL registers.
- Example: MOV M, B

STA 16-bit address

- The contents of the accumulator are copied into the 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.
- Example: STA 4350H

(cont.)

Store accumulator indirect

- STAX Reg. pair The contents of the accumulator are copied into the memory
- location specified by the contents of the operand (register
- pair). The contents of the accumulator are not altered.
- Example: STAX B

Store H and L registers indirect

- SHLD 16-bit address The contents of register L are stored into the memory location
- specified by the 16-bit address in the operand and the contents
- of H register are stored into the next memory location by
- incrementing the operand. The contents of registers HL 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.
- Example: SHLD 2470H

Arithmetic Operations Related to 16 Bits or Register Pairs

Increment register pair by 1

- INX R The contents of the designated register pair are incremented
- by 1 and the result is stored in the same place.
- Example: INX H

Decrement register pair by 1

- DCX R The contents of the designated register pair are decremented
- by 1 and the result is stored in the same place.
- Example: DCX H

Arithmetic Operations Related to Memory

Add memory

- ADD M : The contents of the operand (memory) are added to the contents of the accumulator and the result is stored in the accumulator. The operand is a memory location, its location is specified by the contents of the HL
- registers. All flags are modified to reflect the result of the
- addition.

Subtract memory

- SUB M : The contents of the operand (memory) are subtracted to the contents of the accumulator and the result is stored in the accumulator. The operand is a memory location, its location is specified by the contents of the HL
- registers. All flags are modified to reflect the result of the Subtarction.

Increment memory by 1/Decrement memory by 1

- INR M/DCR M : The contents of the memory are incremented by 1 using INR and decremented by 1 using DCR and the result is stored in the same place. The operand is a memory location, its location is specified by the contents of the HL registers.

Logic Operations : ROTATE and COMPARE

- ROTATE

Rotate accumulator left

- RLC none Each binary bit of the accumulator is rotated left by one
- position. Bit D7 is placed in the position of D0 as well as in
- the Carry flag. CY is modified according to bit D7. S, Z, P,
- AC are not affected.
- Example: RLC

Rotate accumulator right

- RRC none Each binary bit of the accumulator is rotated right by one
- position. Bit D0 is placed in the position of D7 as well as in
- the Carry flag. CY is modified according to bit D0. S, Z, P,
- AC are not affected.
- Example: RRC

Logic Operations : ROTATE (Cont.)

Rotate a

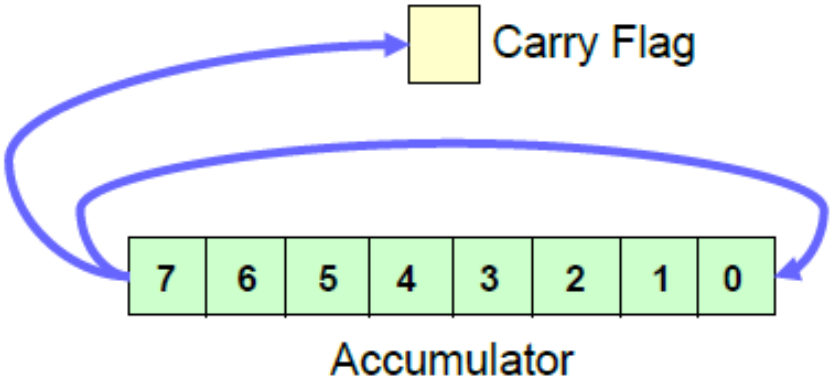
- RAL
- position through the Carry flag. Bit D7 is placed in the Carry
- flag, and the Carry flag is placed in the least significant
- position D0. CY is modified according to bit D7. S, Z, P, AC
- are not affected.
- Example: RAL

Rotate accumulator right through carry

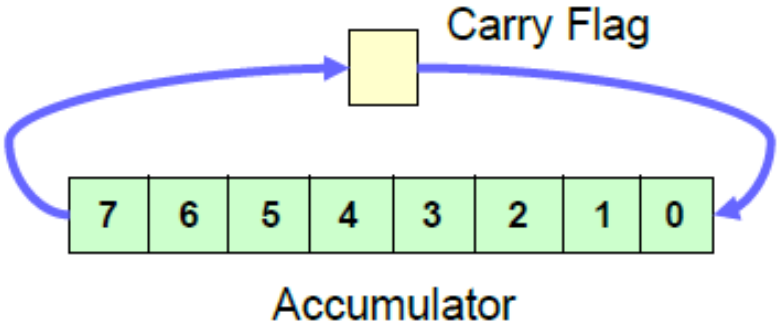
- RAR none Each binary bit of the accumulator is rotated right by one
- position through the Carry flag. Bit D0 is placed in the Carry
- flag, and the Carry flag is placed in the most significant
- position D7. CY is modified according to bit D0. S, Z, P, AC
- are not affected.
- Example: RAR

RLC vs. RLA

- RLC



- RAL



Logical Operations

- Compare
  - Compare the contents of a register or memory location with the contents of the accumulator.
    - CMP R/M — Compare the contents of the register or memory location to the contents of the accumulator.
    - CPI # — Compare the 8-bit number to the contents of the accumulator.
  - The compare instruction sets the flags (Z, Cy, and S).
  - The compare is done using an internal subtraction that does not change the contents of the accumulator.

    A – (R / M / #)

Branch Operations

- Two types:
  - Unconditional branch.
    - Go to a new location no matter what.
  - Conditional branch.
    - Go to a new location if the condition is true.

Unconditional Branch

- JMP Address
  - Jump to the address specified (Go to).
- CALL Address
  - Jump to the address specified but treat it as a subroutine.
- RET
  - Return from a subroutine.

Conditional Branch

- Go to new location if a specified condition is met.
  - JZ Address (Jump on Zero)
    - Go to address specified if the **Zero flag is set**.
  - JNZ Address (Jump on NOT Zero)
    - Go to address specified if the **Zero flag is not set**.
  - JC Address (Jump on Carry)
    - Go to the address specified if the **Carry flag is set**.
  - JNC Address (Jump on No Carry)
    - Go to the address specified if the **Carry flag is not set**.
  - JP Address (Jump on Plus)
    - Go to the address specified if the **Sign flag is not set**
  - JM Address (Jump on Minus)
    - Go to the address specified if the **Sign flag is set**.

Data Formats

- In an 8-bit microprocessor, data can be represented in one of four formats:
  - ASCII
  - BCD
  - Signed Integer
  - Unsigned Integer.

  - It is important to recognize that the microprocessor deals with 0's and 1's.
    - It deals with values as strings of bits.
    - It is the job of the user to add a meaning to these strings.

DATA FORMATS

- Assume the accumulator contains the following value: 0100 0001.
  - There are four ways of reading this value:
    - It is an unsigned integer expressed in binary, the equivalent decimal number would be 65.
    - It is a number expressed in BCD (Binary Coded Decimal) format. That would make it, 41.
    - It is an ASCII representation of a letter. That would make it the letter A.
    - It is a string of 0's and 1's where the 0^{th} and the 6^{th} bits are set to 1 while all other bits are set to 0.

ASCII stands for American Standard Code for Information Interchange.