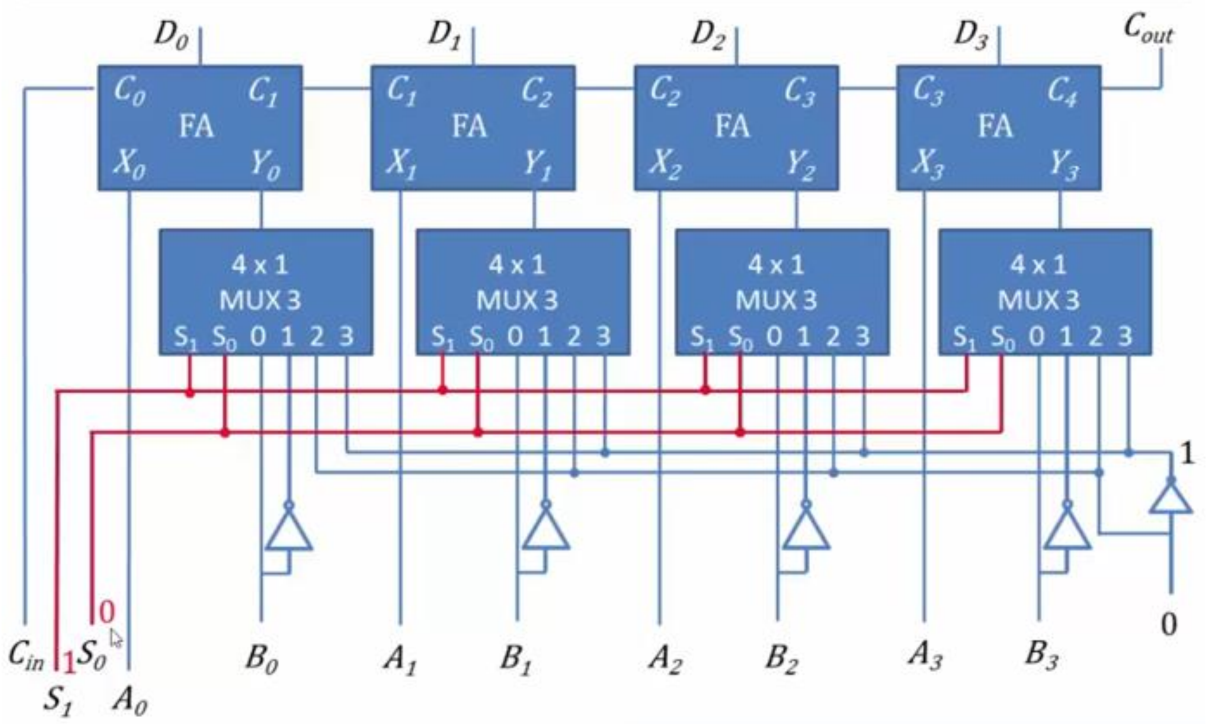
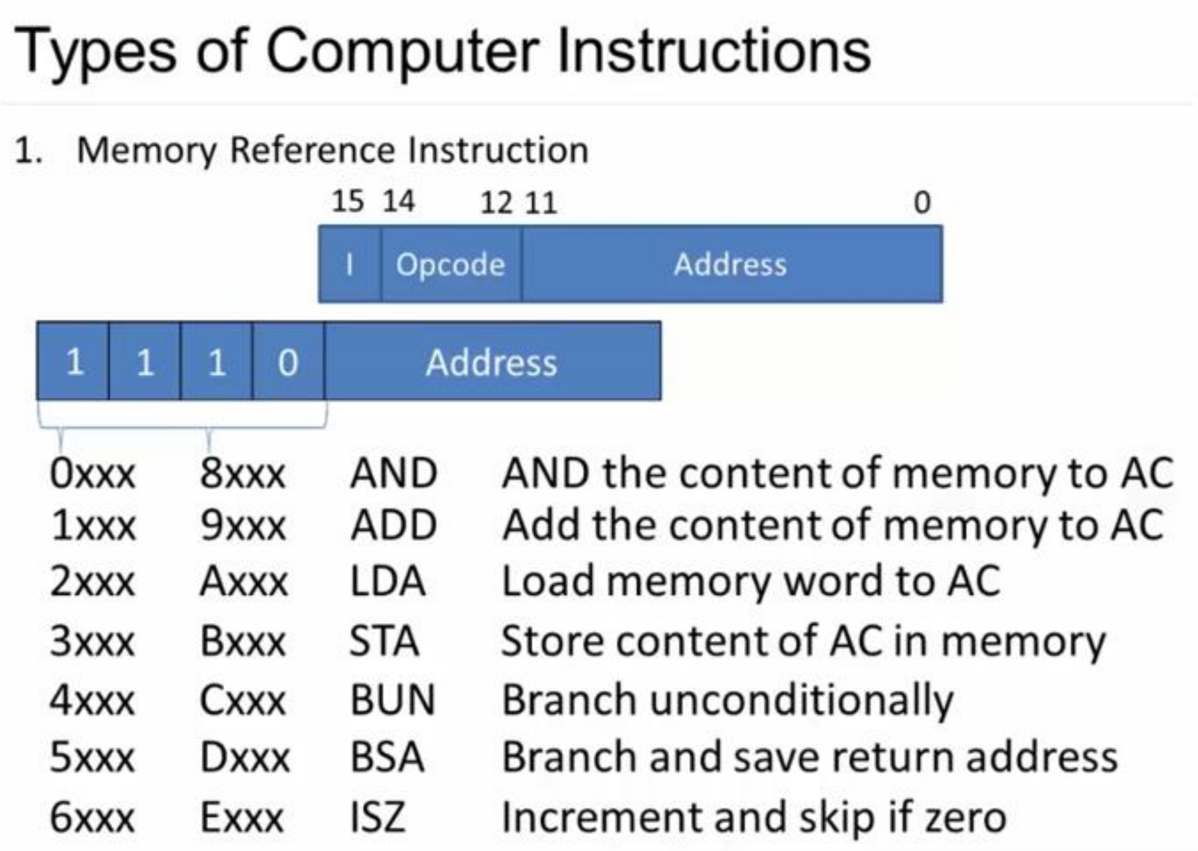
# 4 – bit Arithmetic Circuit



# 4 – bit Arithmetic Circuit

- Arithmetic Circuit Function

| $S_1$ | $S_0$ | $C_{in}$ | Y | $D=A+Y+C_{in}$ | Microoperation |
|---|---|---|---|---|---|
| 0 | 0 | 0 | $B$ | $D = A + B$ | Add |
| 0 | 0 | 1 | $B$ | $D = A + B + 1$ | Add with carry |
| 0 | 1 | 0 | $B'$ | $D = A + B'$ | Subtract with borrow |
| 0 | 1 | 1 | $B'$ | $D = A + B' + 1$ | Subtract |
| 1 | 0 | 0 | 0 | $D = A$ | Transfer |
| 1 | 0 | 1 | 0 | $D = A + 1$ | Increment A |
| 1 | 1 | 0 | 1 | $D = A - 1$ | Decrement A |
| 1 | 1 | 1 | 1 | $D = A$ | Transfer A |



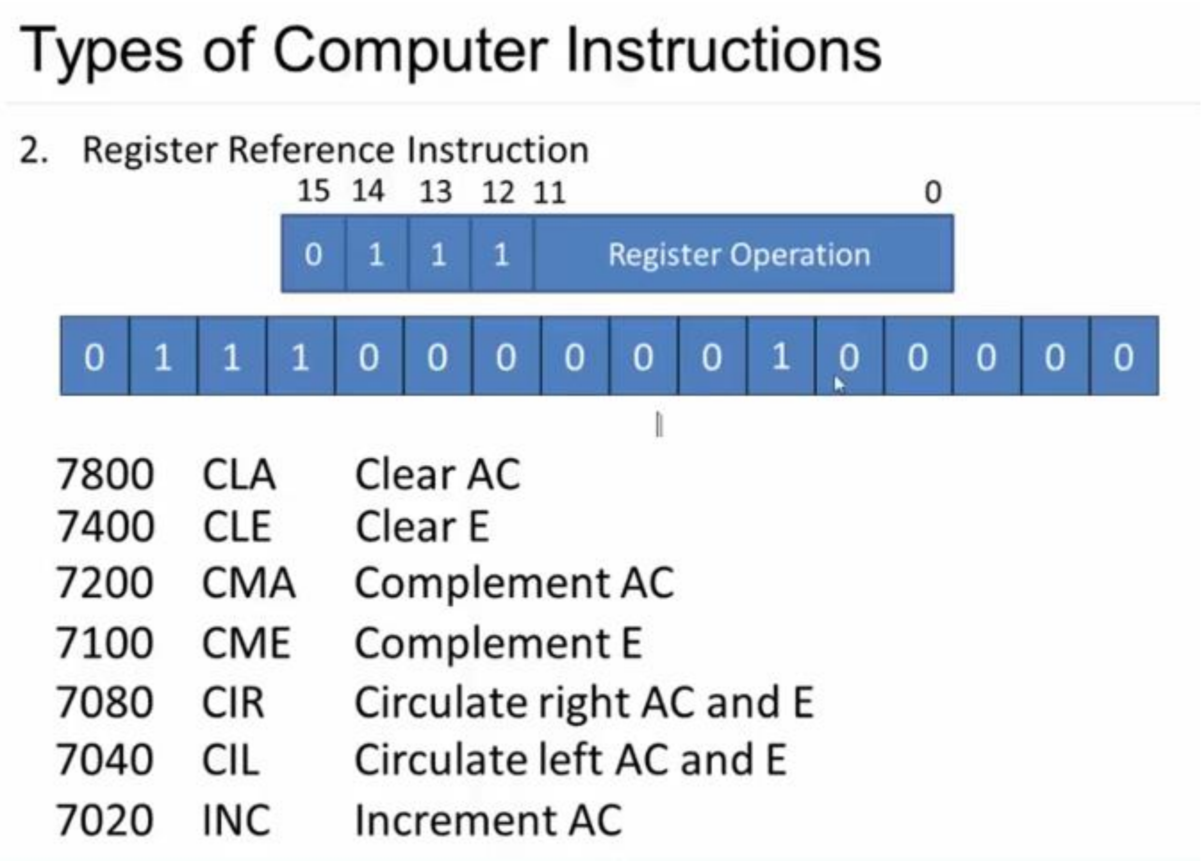
# Types of Computer Instructions

1. Memory Reference Instruction

| 15 | 14 12 | 11 0 |
|---|---|---|
| I | Opcode | Address |

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | Address |

| | | | |
|---|---|---|---|
| 0xxx | 8xxx | AND | AND the content of memory to AC |
| 1xxx | 9xxx | ADD | Add the content of memory to AC |
| 2xxx | Axxx | LDA | Load memory word to AC |
| 3xxx | Bxxx | STA | Store content of AC in memory |
| 4xxx | Cxxx | BUN | Branch unconditionally |
| 5xxx | Dxxx | BSA | Branch and save return address |
| 6xxx | Exxx | ISZ | Increment and skip if zero |



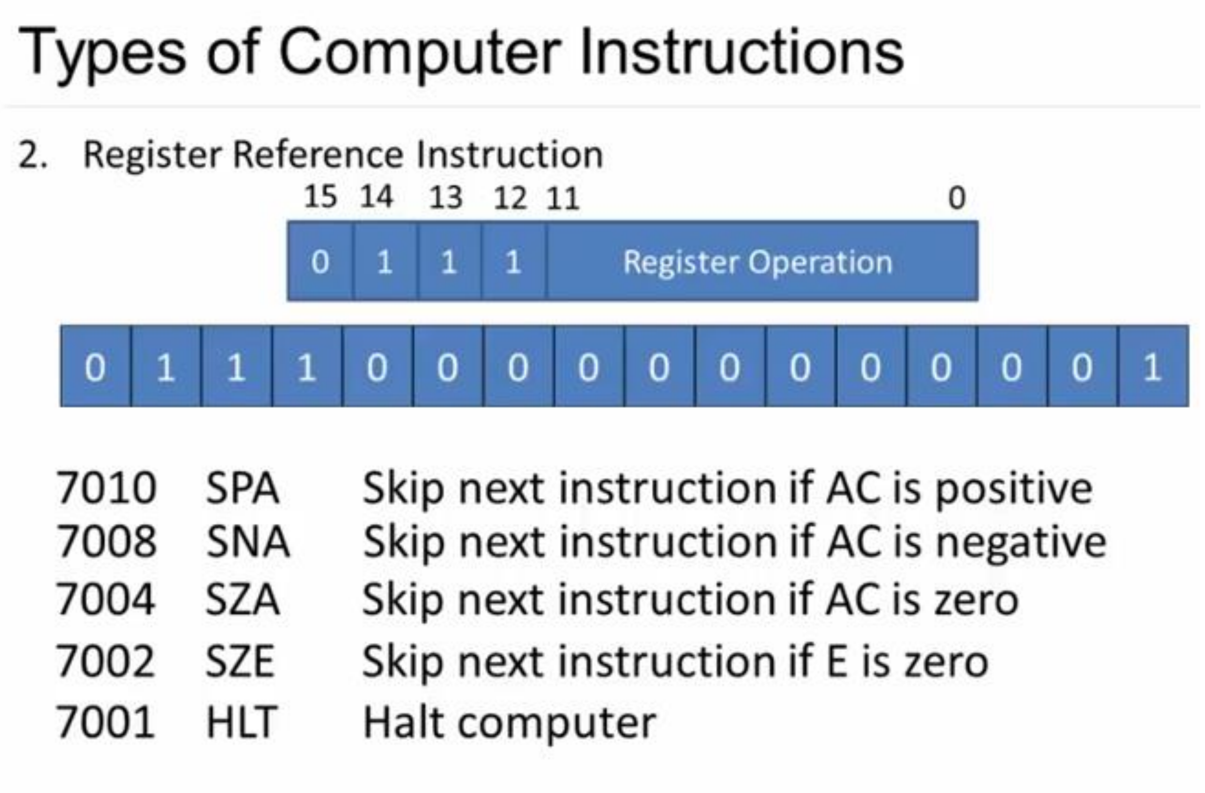
# Types of Computer Instructions

2. Register Reference Instruction

| 15 | 14 | 13 | 12 | 11 0 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | Register Operation |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| 7800 | CLA | Clear AC |
| 7400 | CLE | Clear E |
| 7200 | CMA | Complement AC |
| 7100 | CME | Complement E |
| 7080 | CIR | Circulate right AC and E |
| 7040 | CIL | Circulate left AC and E |
| 7020 | INC | Increment AC |



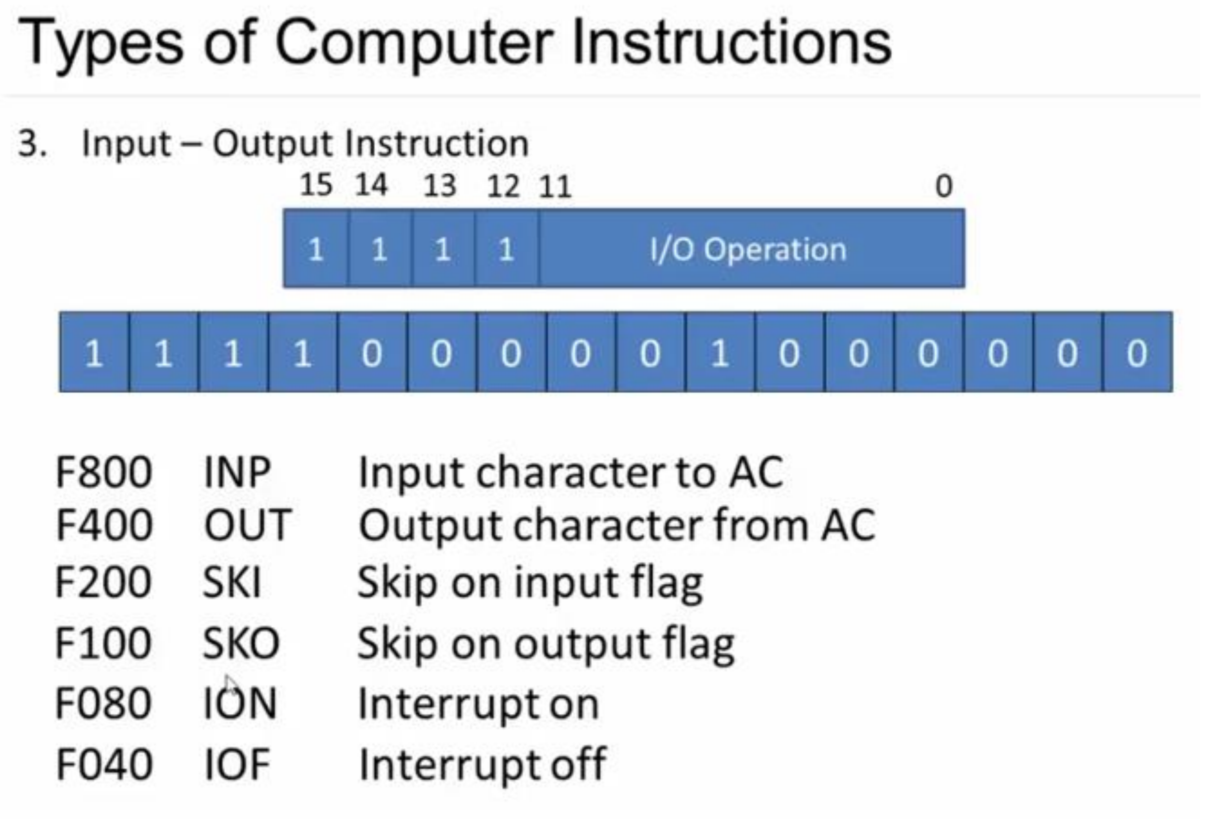
# Types of Computer Instructions

2. Register Reference Instruction

| 15 | 14 | 13 | 12 | 11 … 0 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | Register Operation |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | | |
|---|---|---|
| 7010 | SPA | Skip next instruction if AC is positive |
| 7008 | SNA | Skip next instruction if AC is negative |
| 7004 | SZA | Skip next instruction if AC is zero |
| 7002 | SZE | Skip next instruction if E is zero |
| 7001 | HLT | Halt computer |



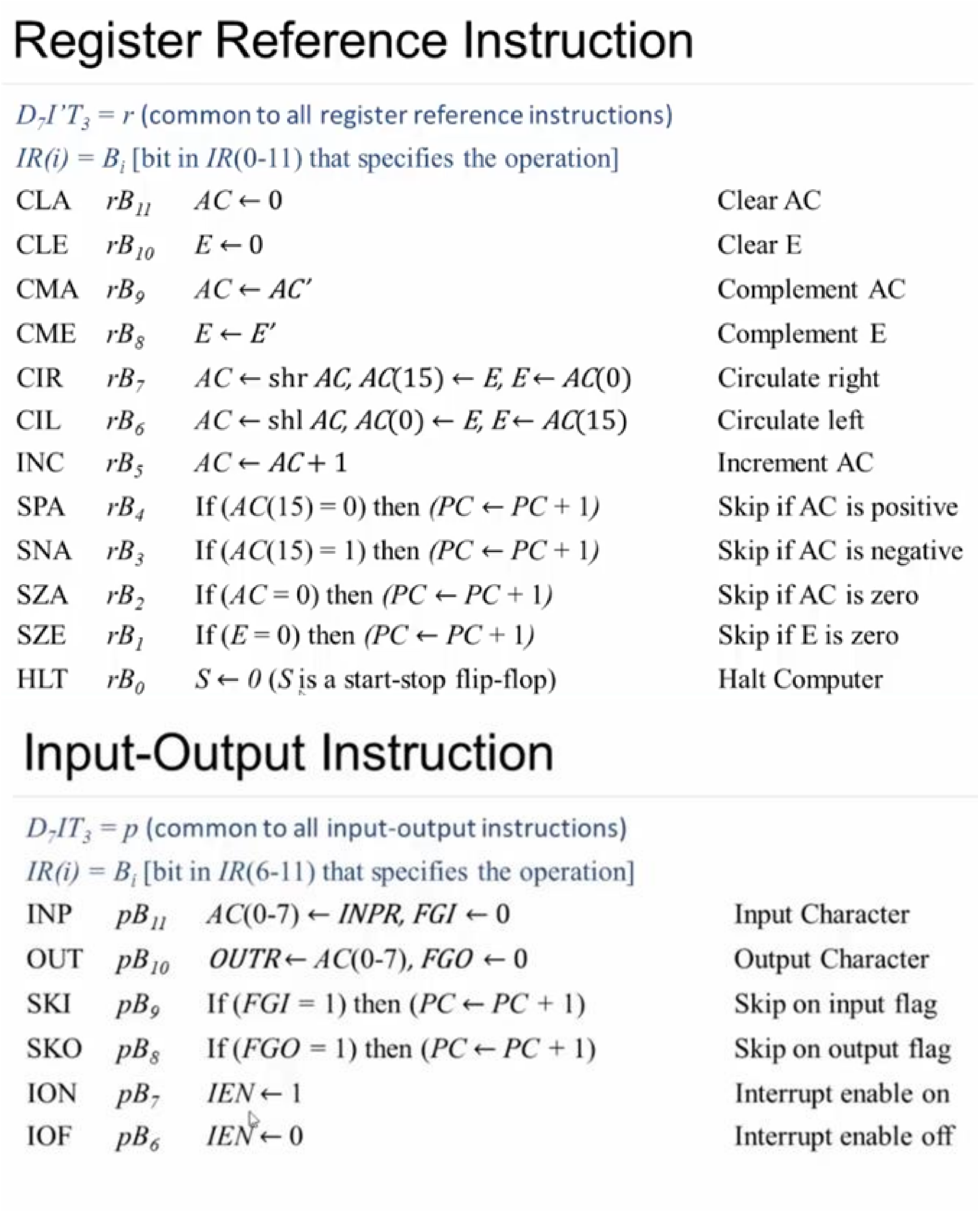
# Types of Computer Instructions

3. Input – Output Instruction

| 15 | 14 | 13 | 12 | 11 … 0 |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | I/O Operation |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | |
|---|---|---|
| F800 | INP | Input character to AC |
| F400 | OUT | Output character from AC |
| F200 | SKI | Skip on input flag |
| F100 | SKO | Skip on output flag |
| F080 | ION | Interrupt on |
| F040 | IOF | Interrupt off |



## Register Reference Instruction

$D_7 I' T_3 = r$ (common to all register reference instructions)

$IR(i) = B_i$ [bit in $IR$(0-11) that specifies the operation]

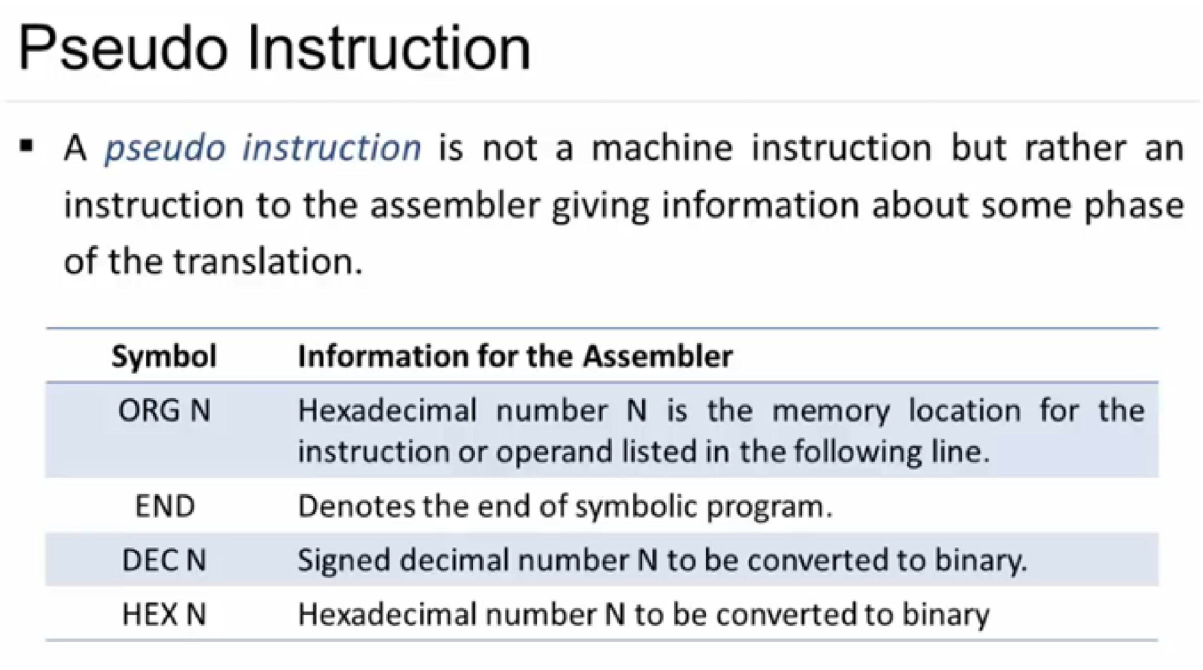
| | | | |
|---|---|---|---|
| CLA | $rB_{11}$ | $AC \leftarrow 0$ | Clear AC |
| CLE | $rB_{10}$ | $E \leftarrow 0$ | Clear E |
| CMA | $rB_9$ | $AC \leftarrow AC'$ | Complement AC |
| CME | $rB_8$ | $E \leftarrow E'$ | Complement E |
| CIR | $rB_7$ | $AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$ | Circulate right |
| CIL | $rB_6$ | $AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$ | Circulate left |
| INC | $rB_5$ | $AC \leftarrow AC + 1$ | Increment AC |
| SPA | $rB_4$ | If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$ | Skip if AC is positive |
| SNA | $rB_3$ | If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$ | Skip if AC is negative |
| SZA | $rB_2$ | If $(AC = 0)$ then $(PC \leftarrow PC + 1)$ | Skip if AC is zero |
| SZE | $rB_1$ | If $(E = 0)$ then $(PC \leftarrow PC + 1)$ | Skip if E is zero |
| HLT | $rB_0$ | $S \leftarrow 0$ ($S$ is a start-stop flip-flop) | Halt Computer |

## Input-Output Instruction

$D_7 I T_3 = p$ (common to all input-output instructions)

$IR(i) = B_i$ [bit in $IR$(6-11) that specifies the operation]

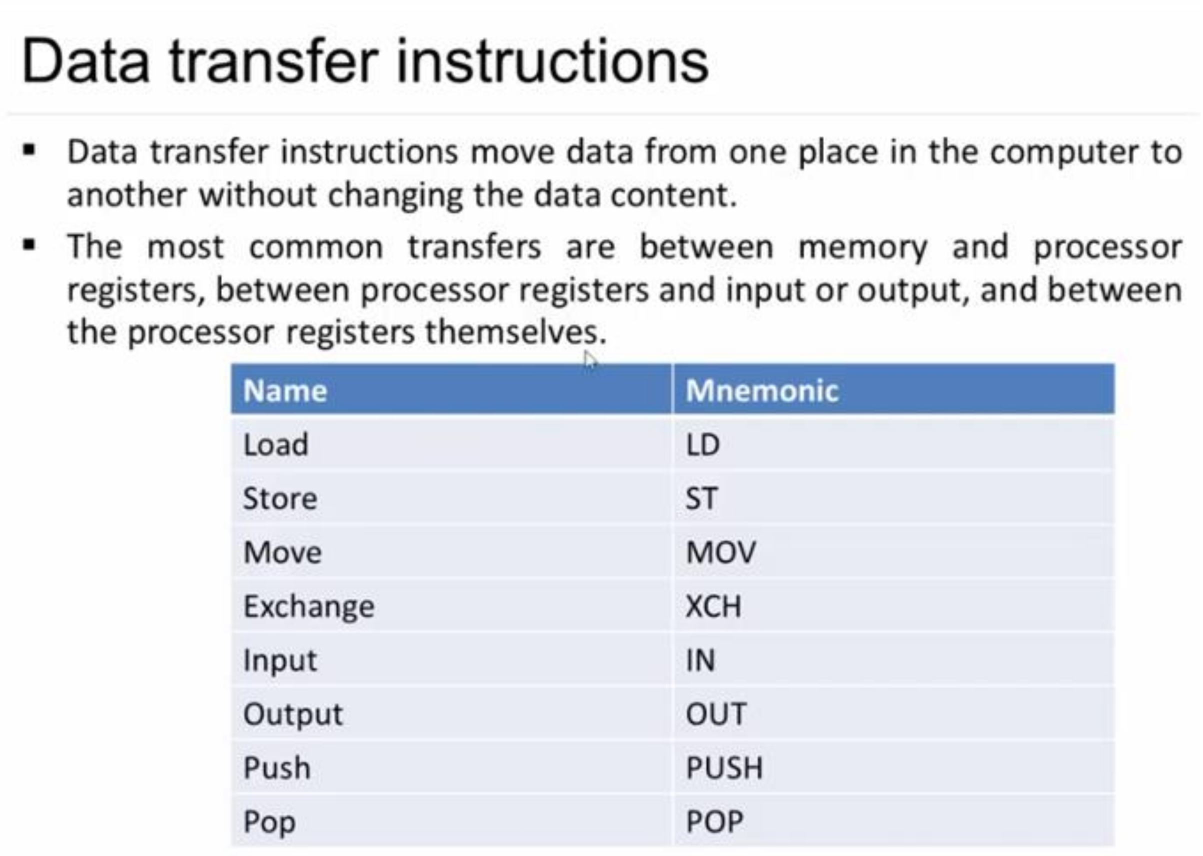
| | | | |
|---|---|---|---|
| INP | $pB_{11}$ | $AC(0\text{-}7) \leftarrow INPR, FGI \leftarrow 0$ | Input Character |
| OUT | $pB_{10}$ | $OUTR \leftarrow AC(0\text{-}7), FGO \leftarrow 0$ | Output Character |
| SKI | $pB_9$ | If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$ | Skip on input flag |
| SKO | $pB_8$ | If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$ | Skip on output flag |
| ION | $pB_7$ | $IEN \leftarrow 1$ | Interrupt enable on |
| IOF | $pB_6$ | $IEN \leftarrow 0$ | Interrupt enable off |



# Pseudo Instruction

- A *pseudo instruction* is not a machine instruction but rather an instruction to the assembler giving information about some phase of the translation.

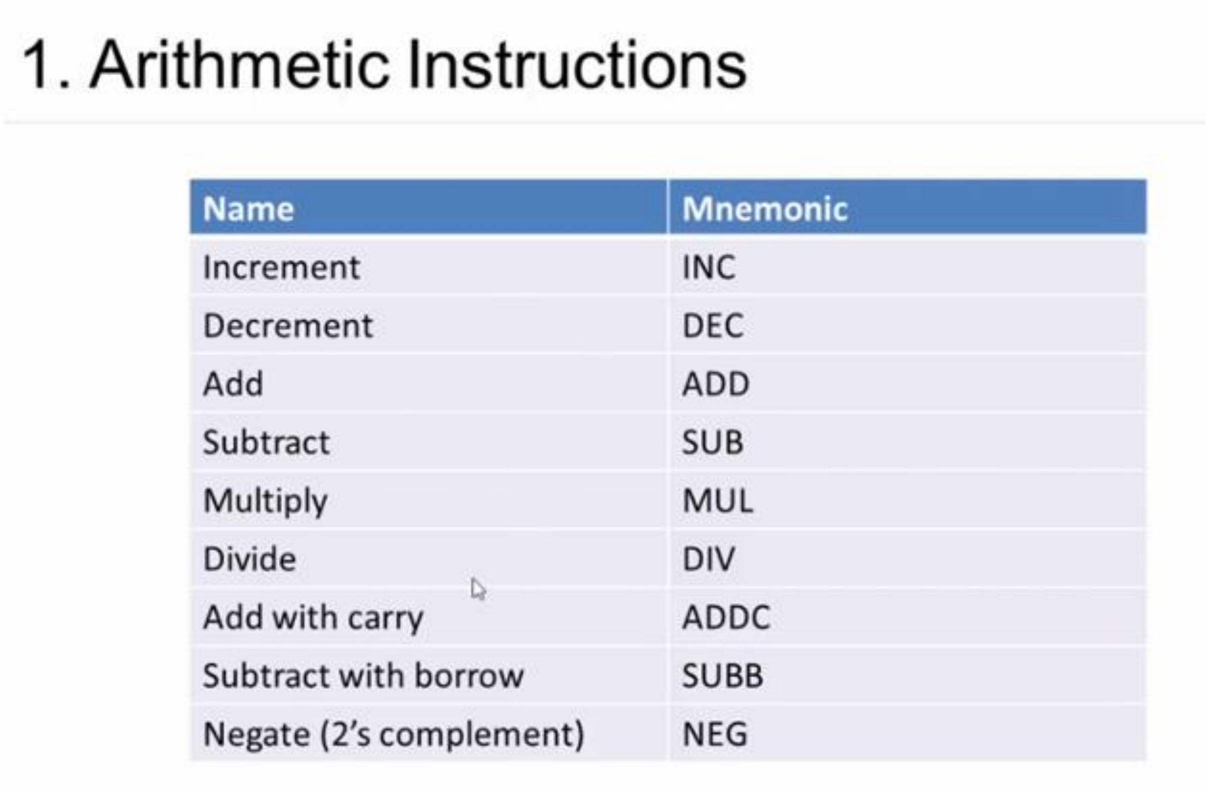
| Symbol | Information for the Assembler |
|---|---|
| ORG N | Hexadecimal number N is the memory location for the instruction or operand listed in the following line. |
| END | Denotes the end of symbolic program. |
| DEC N | Signed decimal number N to be converted to binary. |
| HEX N | Hexadecimal number N to be converted to binary |



# Data transfer instructions

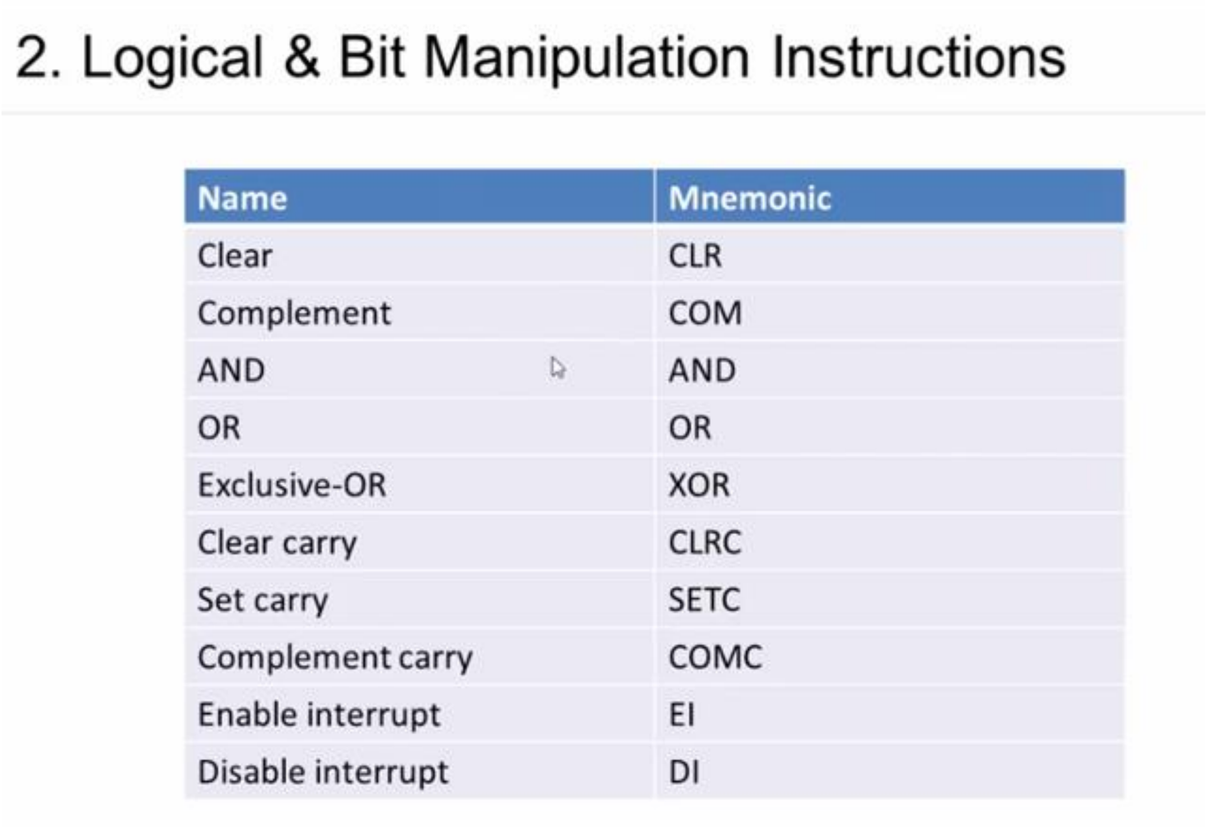
- Data transfer instructions move data from one place in the computer to another without changing the data content.
- The most common transfers are between memory and processor registers, between processor registers and input or output, and between the processor registers themselves.

| Name | Mnemonic |
|---|---|
| Load | LD |
| Store | ST |
| Move | MOV |
| Exchange | XCH |
| Input | IN |
| Output | OUT |
| Push | PUSH |
| Pop | POP |



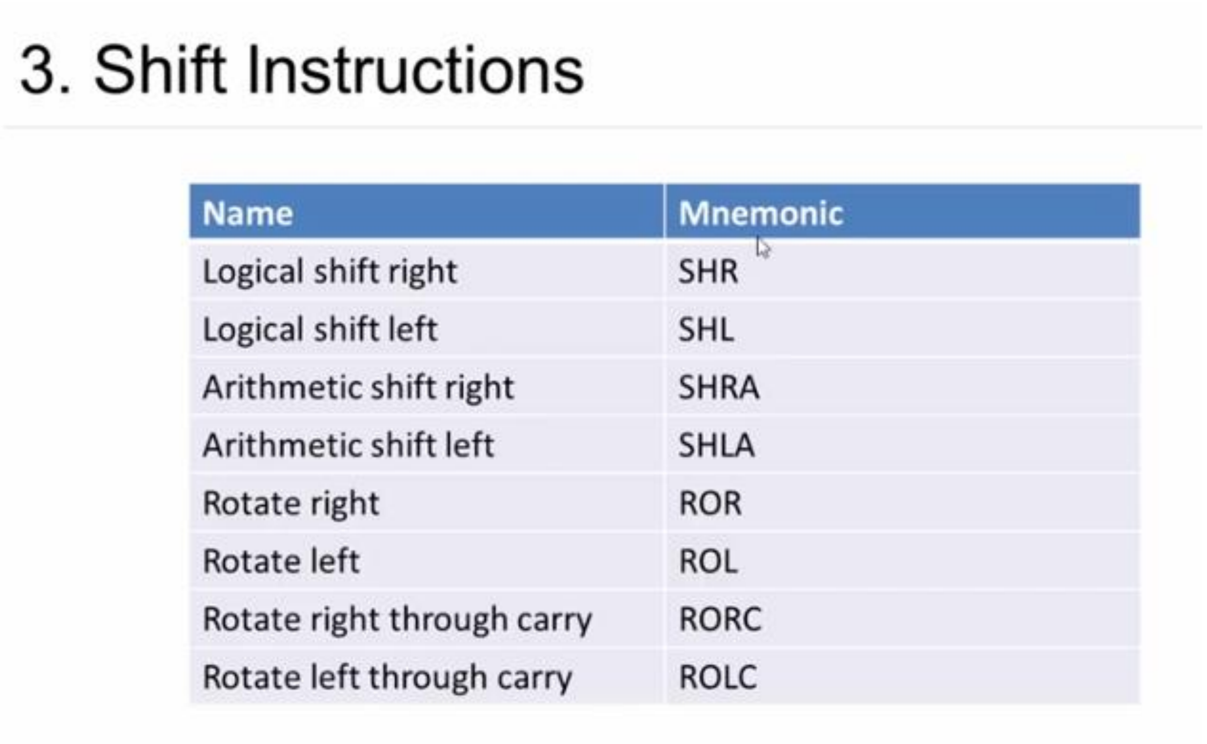
# 1. Arithmetic Instructions

| Name | Mnemonic |
|---|---|
| Increment | INC |
| Decrement | DEC |
| Add | ADD |
| Subtract | SUB |
| Multiply | MUL |
| Divide | DIV |
| Add with carry | ADDC |
| Subtract with borrow | SUBB |
| Negate (2's complement) | NEG |



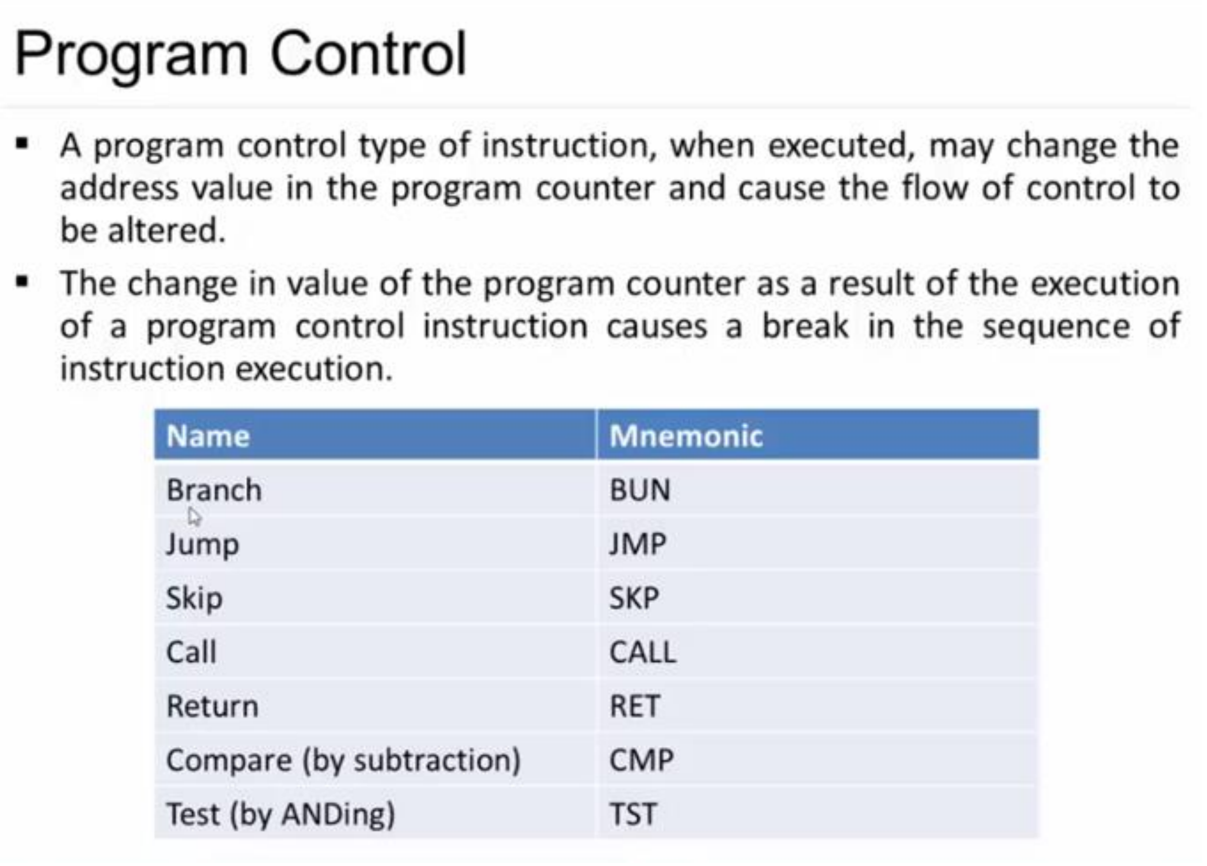
# 2. Logical & Bit Manipulation Instructions

| Name | Mnemonic |
|---|---|
| Clear | CLR |
| Complement | COM |
| AND | AND |
| OR | OR |
| Exclusive-OR | XOR |
| Clear carry | CLRC |
| Set carry | SETC |
| Complement carry | COMC |
| Enable interrupt | EI |
| Disable interrupt | DI |



# 3. Shift Instructions

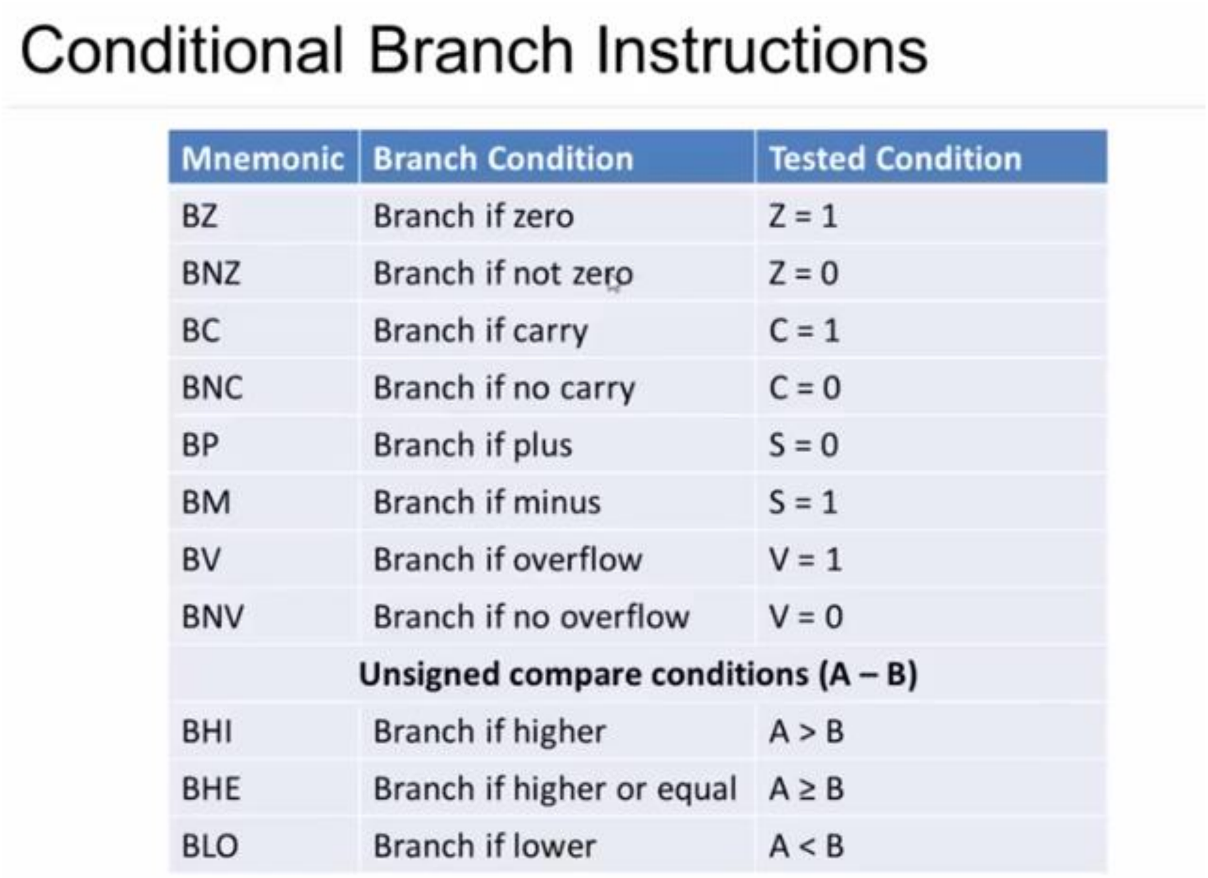
| Name | Mnemonic |
|---|---|
| Logical shift right | SHR |
| Logical shift left | SHL |
| Arithmetic shift right | SHRA |
| Arithmetic shift left | SHLA |
| Rotate right | ROR |
| Rotate left | ROL |
| Rotate right through carry | RORC |
| Rotate left through carry | ROLC |



# Program Control

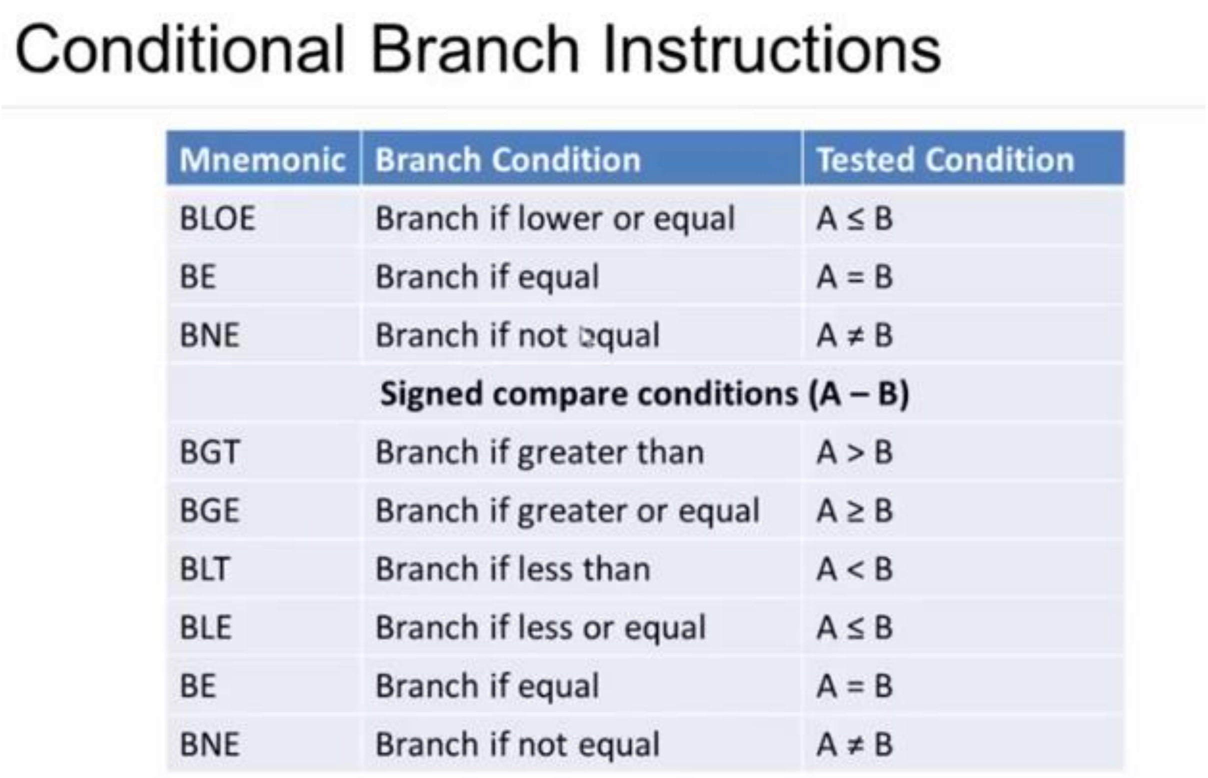
- A program control type of instruction, when executed, may change the address value in the program counter and cause the flow of control to be altered.
- The change in value of the program counter as a result of the execution of a program control instruction causes a break in the sequence of instruction execution.

| Name | Mnemonic |
|---|---|
| Branch | BUN |
| Jump | JMP |
| Skip | SKP |
| Call | CALL |
| Return | RET |
| Compare (by subtraction) | CMP |
| Test (by ANDing) | TST |



# Conditional Branch Instructions

| Mnemonic | Branch Condition | Tested Condition |
|---|---|---|
| BZ | Branch if zero | Z = 1 |
| BNZ | Branch if not zero | Z = 0 |
| BC | Branch if carry | C = 1 |
| BNC | Branch if no carry | C = 0 |
| BP | Branch if plus | S = 0 |
| BM | Branch if minus | S = 1 |
| BV | Branch if overflow | V = 1 |
| BNV | Branch if no overflow | V = 0 |
| | **Unsigned compare conditions (A – B)** | |
| BHI | Branch if higher | A > B |
| BHE | Branch if higher or equal | A ≥ B |
| BLO | Branch if lower | A < B |



# Conditional Branch Instructions

| Mnemonic | Branch Condition | Tested Condition |
|---|---|---|
| BLOE | Branch if lower or equal | A ≤ B |
| BE | Branch if equal | A = B |
| BNE | Branch if not equal | A ≠ B |
| | **Signed compare conditions (A – B)** | |
| BGT | Branch if greater than | A > B |
| BGE | Branch if greater or equal | A ≥ B |
| BLT | Branch if less than | A < B |
| BLE | Branch if less or equal | A ≤ B |
| BE | Branch if equal | A = B |
| BNE | Branch if not equal | A ≠ B |