# Computer Structure and Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran



## RR Instructions:


**Load Register**

Mnemonic: LR r1,r2
Operation: r1 ← (r2);
OPCODE: 18h

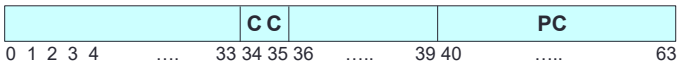
**Example 1:**
Assembly instruction: LR 3,5
Operation: R3 ← (R5);
Machine code: 1835

**Example 2:**
Assembly instruction: LR B'101',X'D'
Operation: R5 ← (R13);
Machine code: 185D

# Condition Code (CC):

All arithmetic and logic instructions update CC (bits 34 and 35 in PSW).

PSW




CC is used with conditional branch instructions (we will see).

It is updated according to the result generated after execution of each instruction as follows:

CC ← 00; if the result = 0
CC ← 01; if the result < 0
CC ← 10; if the result > 0
CC ← 11; on overflow



# RR Instructions:


**Load Complement Register**

Mnemonic: LCR r1,r2
Operation: r1 ← - (r2); and update CC accordingly;
OPCODE: 13h

**Example 1:**
Assembly instruction: LCR 2,2 assume (R2) = -33
Operation: R2 ← 33; and CC ← 10;
Machine code: 1322

**Example 2:**
Assembly instruction: REG5 EQU 5
LCR REG5,1 assume (R1) = 0
Operation: R5 ← 0; and CC ← 00;
Machine code: 1351

# RR Instructions:


**Load Negative Register**

Mnemonic: LNR r1,r2
Operation: r1 ← - |(r2)|; and update CC accordingly;
OPCODE: 11h

**Example 1:**
Assembly instruction: LNR 2,4 assume (R4) = -123
Operation: R2 ← -123; and CC ← 01;
Machine code: 1124

**Example 2:**
Assembly instruction: LNR 5,5 assume (R5) = 13
Operation: R5 ← -13; and CC ← 01;
Machine code: 1155



# RR Instructions:


**Load Positive Register**

Mnemonic: LPR r1,r2
Operation: r1 ← |(r2)|; and update CC accordingly;
OPCODE: 10h

**Example 1:**
Assembly instruction: LPR B'010',7 assume (R7) = -123
Operation: R2 ← 123; and CC ← 10;
Machine code: 1027

**Example 2:**
Assembly instruction: LPR 5,B'101' assume (R5) = 13
Operation: R5 ← 13; and CC ← 10;
Machine code: 1055

# RR Instructions:


**Add Register**

Mnemonic: AR r1,r2
Operation: r1 ← (r1) + (r2); and update CC accordingly;
OPCODE: 1Ah

**Example 1:**
Assembly instruction: AR 2,7 assume (R2) = -123, (R7) = 5
Operation: R2 ← -118; and CC ← 01;
Machine code: 1A27

**Example 2:**
Assembly instruction: AR 5,6 assume (R5) = -13, (R6) = 20
Operation: R5 ← 7; and CC ← 10;
Machine code: 1A56



# RR Instructions:


**Subtract Register**

Mnemonic: SR r1,r2
Operation: r1 ← (r1) - (r2); and update CC accordingly;
OPCODE: 1Bh

**Example 1:**
Assembly instruction: SR 8,8 assume (R8) = -50000
Operation: R8 ← 0; and CC ← 00;
Machine code: 1B88

**Example 2:**
Assembly instruction: SR 5,6 assume (R5) = -13, (R6) = -22
Operation: R5 ← 9; and CC ← 10;
Machine code: 1B56

# RR Instructions:


**Multiply Register**

Mnemonic: MR r1,r2 r1 must be an even numbered register
Operation: r1:r1+1 ← (r1+1)*(r2); and update CC accordingly;
OPCODE: 1Ch

**Example 1:**
Assembly instruction: MR 8,4 assume (R8)=-5, (R9)=5, (R4)=-7
Operation: R8:R9 ← -35; and CC ← 01; ➔ (R8)=-1, (R9)=-35
Machine code: 1C84

**Example 2:**
Assembly instruction: MR 4,5 assume (R4)=-10, (R5)=-2
Operation: R4:R5 ← 4; and CC ← 10; ➔ (R4)=0, (R5)=4
Machine code: 1C45



# RR Instructions:


**Divide Register**

Mnemonic: DR r1,r2 r1 must be an even numbered register
Operation: r1+1 ← $\frac{(r1):(r1+1)}{(r2)}$; r1 ← remainder; update CC accordingly;
OPCODE: 1Dh

**Example 1:**
Assembly instruction: DR 8,4 assume (R8):(R9)=503, (R4)=5
Operation: R9 ←100; R8 ←3; and CC ← 10;
Machine code: 1D84

**Example 2:**
Assembly instruction: DR 4,6 assume (R4):(R5) =-11, (R6)=3
Operation: R5 ← -3; R4 ← -2; and CC ← 01;
Machine code: 1D46

# RR Instructions:


**Add Logical Register**

Mnemonic: ALR r1,r2 Pure binary addition
Operation: r1 ← (r1) + (r2); and update CC accordingly;
OPCODE: 1Eh

**Example 1:**
Assembly instruction: ALR 2,7 assume (R2) = FFFFFFFEh, (R7) = 5
Operation: R2 ← 3; and CC ← 11;
Machine code: 1E27

**Example 2:**
Assembly instruction: ALR 5,6 assume (R5) = 13, (R6) = 20
Operation: R5 ← 33; and CC ← 10;
Machine code: 1E56



# RR Instructions:


**Subtract Logical Register**

Mnemonic: SLR r1,r2 Pure binary subtract
Operation: r1 ← (r1) - (r2); and update CC accordingly;
OPCODE: 1Fh

**Example 1:**
Assembly instruction: SLR 8,8 assume (R8) = FFF78955h
Operation: R8 ← 0; and CC ← 00;
Machine code: 1F88

**Example 2:**
Assembly instruction: SLR 5,6 assume (R5) = 20, (R6) = 32
Operation: R5 ← FFFFFFF4h; and CC ← 11;
Machine code: 1F56

# RR Instructions:


**And Register**

Mnemonic: NR r1,r2 Bitwise Logical And
Operation: r1 ← (r1) ∧ (r2); and update CC accordingly;
OPCODE: 14h

**Example 1:**
Assembly instruction: NR 4,9 assume (R4)=FFFFFFFEh, (R9) = 1
Operation: R4 ← 0; and CC ← 00;
Machine code: 1449 00…00 = 11…110 ∧ 00…01

**Example 2:**
Assembly instruction: NR 5,6 assume (R5) = 20, (R6) = 36
Operation: R5 ← 4; and CC ← 10;
Machine code: 1456
00…00100 = 00…010100 ∧ 00…0100100



# RR Instructions:


**Or Register**

Mnemonic: OR r1,r2 Bitwise Logical Or
Operation: r1 ← (r1) ∨ (r2); and update CC accordingly;
OPCODE: 16h

**Example 1:**
Assembly instruction: OR 4,6 assume (R4)=FFFFFFFEh, (R6)=1
Operation: R4 ← -1; and CC ← 01;
Machine code: 1646 11…11 = 11…110 ∨ 00…01

**Example 2:**
Assembly instruction: OR 5,6 assume (R5) = 20, (R6) = 36
Operation: R5 ← 52; and CC ← 10;
Machine code: 1656
00…0110100 = 00…010100 ∧ 00…0100100

# RR Instructions:


**Exclusive-Or Register**

Mnemonic: XR r1,r2 Bitwise Logical Exclusive-Or
Operation: r1 ← (r1) xor (r2); and update CC accordingly;
OPCODE: 17h

**Example 1:**
Assembly instruction: XR 4,6 assume (R4)=FFFFFFFEh, (R6)=11
Operation: R4 ← -11; and CC ← 01;
Machine code: 1746 11…10101 = 11…110 xor 00…01011

**Example 2:**
Assembly instruction: XR 5,6 assume (R5) = -1, (R6) = -10
Operation: R5 ← 9; and CC ← 10;
Machine code: 1756
00…001001 = 11…11111 xor 11…10110



# RR Instructions:


CC ← 00; if (r1)=(r2)
CC ← 01; if (r1)<(r2)
CC ← 10; if (r1)>(r2)

**Compare Register**

Mnemonic: CR r1,r2
Operation: Realize (r1)-(r2), and update CC accordingly;
OPCODE: 19h

**Example 1:**
Assembly instruction: CR 8,8 assume (R8) = -50780
Operation: CC ← 00;
Machine code: 1988

**Example 2:**
Assembly instruction: CR 5,6 assume (R5) = -13, (R6) = -22
Operation: CC ← 10;
Machine code: 1956

# RR Instructions:


CC ← 00; if (r1)=(r2)
CC ← 01; if (r1)<(r2)
CC ← 10; if (r1)>(r2)

**Compare Logical Register**

Mnemonic: CLR r1,r2 Pure binary comparison
Operation: Realize (r1)-(r2) in pure binary, and update CC accordingly;
OPCODE: 15h

**Example 1:**
Assembly instruction: CLR 8,8 assume (R8) = FFF78901h
Operation: CC ← 00;
Machine code: 1588

**Example 2:**
Assembly instruction: CLR 5,6 assume (R5) = -1, (R6) = 31
Operation: CC ← 10;
Machine code: 1556

11…11111 > 00…011111



# RR Instructions:


**Branch on Count Register**

Mnemonic: BCTR r1,r2
Operation: r1 ← (r1) – 1; if (r1)≠0 then PC ← (r2);
OPCODE: 06h

**Example 1:**
Assembly instruction: BCTR 8,7 Assume (R8) = 2, (R7) = 011FFFF0h
Operation: R8 ← 1; PC← (R7);
Machine code: 0687

**Example 2:**
Assembly instruction: BCTR 5,0 Assume (R5) = 23, (R0) = -22
Operation: R5 ← 22;
Machine code: 0650 This is the cheapest way to decrease the content of a register.

## Tricky Facts:

All opcodes are selected carefully and accurately and NOT accidentally, either for reducing complexity or for the sake of design beauty ☺

AR 1Ah
SR 1Bh
MR 1Ch
DR 1Dh
NR 14h
OR 16h
XR 17h



End of Slides