CSE2006
Microprocessor & Interfacing

Module – 1
Introduction to 8086 Microprocessor

Dr. E. Konguvel
Assistant Professor (Sr. Gr. 1),
Dept. of Embedded Technology,
School of Electronics Engineering (SENSE),
konguvel.e@vit.ac.in
9597812810

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Instruction Set

Different Categories of 8086 Instructions:

1. Data Transfer Instructions
2. Push & Pop Instructions
3. Arithmetic Instructions
4. Arithmetic Adjust Instructions
5. Logical & Bit Manipulation Instructions
6. Jump Instructions
7. Loop Instructions
8. Call & Return Instructions
9. String Instructions
10. Processor Control Instructions

Instruction Set

1. Data Transfer Instructions

- To transfer data between registers, registers and memory, registers and immediate data, or memory and immediate data. (both 8 & 16 bit registers)

MOV Destination, Source (Copy data from source to destination)

Destination ← Source, Flag affected: None

Destination	*Source*
Register	Immediate data
Memory	Immediate data
Register	Register
Register	Memory
Memory	Register
Segment	Memory
Memory	Segment
Register	Segment
Segment	Register

Instruction Set

1. Data Transfer Instructions

- MOV instruction perform data movement between registers, registers and memory, registers and immediate data, memory and immediate data, between two memory locations, between I/O port and registers and between the stack and memory or a register.

- The MOV instruction will not be able to
  - set the value of the CS and IP registers
  - copy the value of one segment register to another segment register (should copy to general register first).
  - copy an immediate value to segment register (should copy to general register first)

Instruction Set

1. Data Transfer Instructions

Examples:

i. MOV AL, FFH
ii. MOV [0345], 23H
iii. MOV [0345], 2345H
iv. MOV [BX], 45H.
v. MOV AL, [2340]
vi. MOV [4000], AL
vii. MOV BX, CX

Instruction Set

1. Data Transfer Instructions

XCHG Destination, Source (Exchange data between source to destination)

Destination ↔ Source, Flag affected: None

Destination	*Source*
Accumulator	register
Memory	register
Register	register

- Not Allowed:
  - Contents of two memory locations
  - Immediate data

Examples:

- XCHG [4000], AX
- XCHG AX, BX
- XCHG AL, [BX]

Instruction Set

1. Data Transfer Instructions

LAHF (Loads the lower flags byte into AH)

AH ← Flags, Flag not affected: O A C Z P

AH = flags register

AH bit: 7 6 5 4 3 2 1 0

[SF] [ZF] [0] [AF] [0] [PF] [1] [CF]

SAHF (Saves AH into lower flags byte)

Flags ← AH, Flag affected: None

Saves the AH register bit pattern into the low byte of the flags register.

Instruction Set

1. Data Transfer Instructions

IN port8 or DX (Input data from I/O device)

byte: AL ← port

word: AL ← [port]; AH ← [port+1] or AX ← (DX) Flag affected: None

- To read data from an input port. (Address of the input port can be specified directly or indirectly)
- Destination registers: AL and AX.
- DX is the only register to carry the port address.
- Examples:
  - IN AL, 01
  - IN AX, DX

Instruction Set

1. Data Transfer Instructions

OUT port8 or DX (Output data to I/O device)

byte: [port] ← AL

word: [port] ← AL [port+1]*AH or (DX←AX)

Flag affected: None

- To write on an output port.
- The address of the output port may be specified in the instruction directly or implicitly in DX.
- Examples:
  - OUT 02, AL
  - OUT DX, AX

Instruction Set

1. Data Transfer Instructions

LEA reg16, addr (load effective address)

reg16 ← effective address (offset) of addr. Flag affected: None

- Loads the effective address or offset of memory variable into reg16.
  - LEA BX, [0245]

LDS reg16, memory (load data segment)

reg16 ← [memory16]; DS ← [memory16+2] Flag affected: None

- Loads the DS register and reg16 from memory with the segment and offset values.
  - LDS AX, [BX]

Instruction Set

1. Data Transfer Instructions

LES reg16, memory (Load extra segment)

reg16 ← [mem16]; ES ← [mem16 + 2] Flag affected: None

- Loads the ES register and reg16 with the segment and offset values for the variable in memory.
- LES CX, [4000]

XLAT (Translate byte in AL by table look-up)

AL ← DS: [BX+AL], Flag affected: None

- To translate the byte in the AL register by adding it to a base value in BX which has been set to locate the look-up table and the byte located is returned in AL.

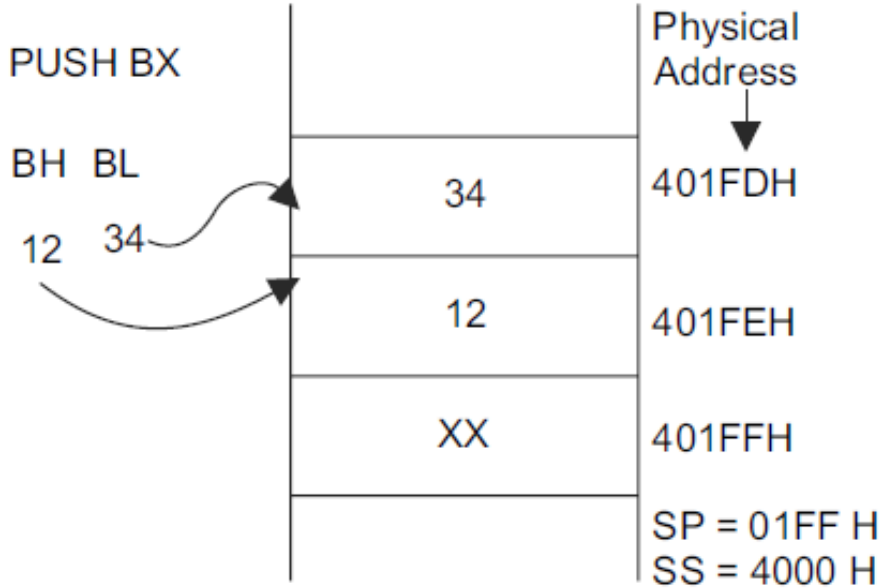
Instruction Set

2. Push & Pop Instructions

PUSH Source Push source register onto stack

SP = SP-2; SS:[SP] ← source register, Flag affected: None

- Content of a specified register is pushed on to the stack.
- SP is decremented by 2 after execution, and then stores the two-byte contents of the operand onto stack.
- PUSH AX
- PUSH BX
- PUSH CX
- PUSH DS
- PUSH [4000]



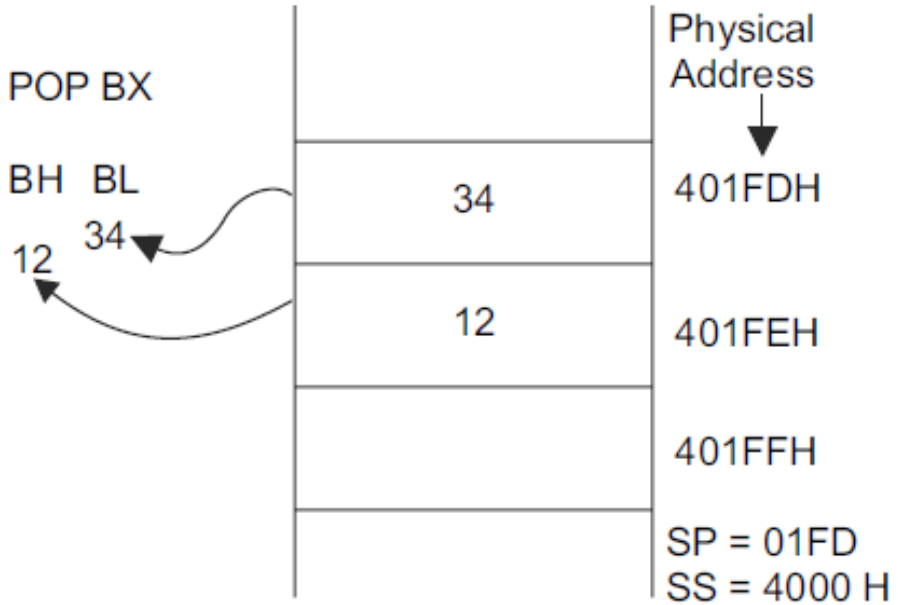
Instruction Set

2. Push & Pop Instructions

POP destination (Pop word at top of stack to destination)

Destination ← SS:[SP]; SP = SP + 2, Flag affected: None

- Loads the specified register/memory location with the contents of current SS and SP.
- SP is incremented by 2.
- POP AX
- POP BX
- POP CX
- POP DS
- POP [4000]



Instruction Set

2. Push & Pop Instructions

PUSHF (Push flags word onto stack)

SP = SP-2; SS: [SP] ← flags, SP = SP-2; SS:[SP] ← Source, Flag affected: None

- Pushes the content of the flag register on the stack, First the upper byte $FLAG_U$ and the lower byte $FLAG_L$.
- SP is decremented by 2 for each push operation.

POPF (Pop word at top of stack to flags register)

flags ← SS:[SP]; SP = SP + 2, Flag affected: All

- POPF – Reverse of PUSHF

Instruction Set

3. Arithmetic Instructions

ADD Destination, Source (Add two operands, result remains in destination)

Destination ← (Source + Destination), Flag affected: O S Z A P C

- Memory to Memory addition is not allowed.
- Examples:
  - ADD AL, 22H; ADD AX, BX; ADD AL, [BX];
  - ADD [BX],CL; ADD [BX],CX

ADC Destination, Source (Add two operands with carry from previous add)

Destination ← (Source + Destination + CF), Flag affected: O S Z A P C

- Examples:
  - ADC AX, 1234H; ADC AX, CX; ADC AX, [SI]; ADC AX, [4000];
  - ADC [SI], AX; and ADC [4000], BX.

Instruction Set

3. Arithmetic Instructions

SUB Destination, Source (Subtract source from destination, store result in destination)

Destination ← (Destination-Source), Flag affected: O S Z A P C

- Examples:
  - SUB AL, 44H; SUB AX, BX; SUB AL, [BX];
  - SUB [BX], CL;SUB [BX], CX.

SBB Destination, Source (Subtract source and the carry flag bit from destination)

Destination ←((Destination−Source)-CF) Flag affected: O S Z A P C

- Examples:
  - SBB AX, BX; SBB AL, [BX]; SBB [BX], CL
  - SBB [BX], CX; SBB AX, [4000].

Instruction Set

3. Arithmetic Instructions

INC Destination (Add 1 to destination)

Destination ← (Destination +1), Flag affected: O S Z A P

- Examples:
  - INC BX; INC CX; INC DX; INC [BX].

DEC Destination (Decrement destination by 1

Destination ← (Destination - 1), Flag affected: O S Z A P C

- Examples:
  - DEC BX; DEC CX; DEC DX; DEC [BX].

Instruction Set

3. Arithmetic Instructions

NEG Destination (Changes the sign of an operand (Negate))

Destination ← (0-Destination), Flag affected: O S Z A P C

- Examples:
  - NEG AX; NEG BX; NEG CX; NEG DX; NEG AL; NEG BL;
  - NEG CL and NEG DL.

CMP Destination, Source (Compare by subtracting source from destination)

Destination-Source; Flag affected: O S Z A P C

- When both the source and destination operands are equal, zero flag is set.
- While the source is greater than destination, carry flag is set; otherwise carry flag is reset.
- Examples:
  - CMP BX, 1234; CMP AL, 22; CMP BX, [SI]; CMP [0100], BX
  - CMP [BX], CX.

Instruction Set

3. Arithmetic Instructions

MUL Source (Multiply 8- or 16-bit source by 8-bit (AL) or 16-bit (AX) value (unsigned)

AX ← (AL * source 8)

DX:AX ←(AX * source16), Flag affected: O C; the S, Z, A, and P flags are left in an indeterminate condition

- Immediate operand is not allowed
- Examples:
  - MUL CL; MUL BX; MUL CX; MUL DX; MUL [BX+10].

IMUL Source (Multiply 8-bit or 16-bit source by 8-bit (AL) or 16-bit (AX) value (signed))

AX ←(AL * source 8)

DX: AX ← (AX * source 16), Flag affected: O C; the S, Z A and P flags are left in an indeterminate condition.

- Examples:
  - IMUL CL; IMUL BH; IMUL BX; IMUL CX; IMUL DX; IMUL [BX+10].

Instruction Set

3. Arithmetic Instructions

DIV Source (Divide of 16-bit or 32-bit number by 8- or 16-bit number (unsigned))

AL ← (AX ÷ Source 8)
AH ← Remainder
AX ← (DX: AX ÷ Source 16)
DX ← Remainder
Flag affected: The O, S, Z, A, P, and C flags are left in an inderterminate condition.

- Examples:
  - DIV CL; DIV BX; DIV CX; DIV DX; DIV [BX+10]

IDIV Source (Divide of signed 16-bit or 32-bit number by 8- or 16-bit number (signed)

AL ← (AX ÷ source 8)
AH ← Remainder
AX ← (DX: AX ÷ source 16)
DX ← Remainder
Flag affected: The O, S, Z, A, P, and C flags are left in an indeterminate condition.

- IDIV BH; IDIV CL; IDIV BX; IDIV CX ; IDIV DX; IDIV [BX+10]

Instruction Set

4. Arithmetic Adjust Instructions

DAA (Decimal adjustment after addition)

AL← (AL adjusted for BCD addition), Flag affected: S Z A P C; the O flag is left in an indeterminate condition.

- After addition, the lower nibble is greater than 9, AF is set. 06 will be added to the lower nibble in AL. If the upper nibble of AL is greater than 9 or if the carry flag is set, 60H will be added.

DAS (Decimal adjust for subtraction)

AL ← (AL adjusted for BCD subtraction); Flag affected S Z A P C; the O flag is left in an indeterminate condition.

- While the lower nibble of AL is greater than 9, 06 will be subtracted from lower nibble of AL. If the result of subtraction sets the carry flag or if the upper nibble is greater than 9, 60H will be subtracted from AL.

Instruction Set

4. Arithmetic Adjust Instructions

AAA (ASCII adjust for addition)

AL ← (AL adjusted for ASCII addition); Flag affected: A, C;
the O, S, Z, and P flags are left in an indeterminate state.

- Lower 4 bits of AL will be checked whether it is a valid BCD number in the range 0 to 9. If the lower 4 bits of AL is between 0 to 9 and AF is zero, AAA sets the higher order 4 bits of AL to 0.
- If the value in the lower 4 bits of AL is greater than 9 then the AL is incremented by 06, AH is incremented by 1.

AAS (ASCII adjust for subtraction)

AL ←(AL adjusted for ASCII subtraction); Flag affected: A C; the O, S, Z, and P flags are left in an indeterminate state.

AAM (ASCII adjust for multiplication)

AH: AL ←(AH: AL adjusted for ASCII multiplication); Flag affected: S Z P; the O, A, and C flags are left in an indeterminate condition.

Instruction Set

4. Arithmetic Adjust Instructions

CBW (Convert from byte to word (16-bit ← 8-bit)

AH← (filled with bit-7 of AL), AX ←(AL * source 8) Flag affected: None

CWD (Convert from word to double word)

DX ← (filled with bit-15 of AX), AX ←(AL * source 8), Flag affected: None

Instruction Set

5. Logical & Bit Manipulation Instructions

NOT Destination (1′s complement of destination)

Destination ← (~Destination), Flag affected: None
Converts 1′s to 0′s and 0′s to 1′s in destination.

NOT BL; NOT CL; NOT DL; NOT AX; NOT BX; NOT CX; NOT [BX]

AND Destination, Source (Logical AND)

Destination ← (Destination AND Source)

AND AX, BX; AND CX, DX; AND AX, [BX]; AND AX, [SI]

OR Destination, Source (Logical OR)

Destination ←(Destination OR Source)

OR AX, BX; OR CX, DX; OR AX, [BX]; OR AX, [SI]

Instruction Set

5. Logical & Bit Manipulation Instructions

XOR Destination, Source (Exclusive logical OR)

Destination ←(Destination XOR Source)

XOR AX, BX; XOR CX, DX; XOR AX, [BX] and XOR AX, [SI]

TEST Destination, Source (Non-destructive logical AND)

Flags ← (Destination AND Source)

TEST AX, BX; TEST CX, DX; TEST AX, [BX]; TEST AX, [DI]

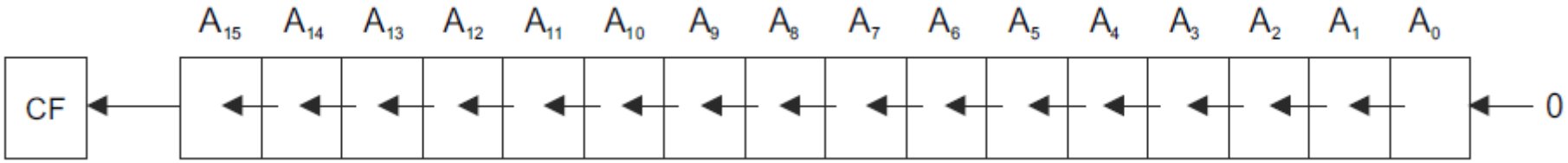
Instruction Set

5. Logical & Bit Manipulation Instructions

SHL/SAL (Shift Logical/ Arithmetic Left)

$A_{n+1} \leftarrow A_n$, $A_{15} \leftarrow A_{14}$, $A_0 \leftarrow 0$, $CF \leftarrow A_{15}$, All flags are affected

- SHL AX, CL
- SHL AX, 1



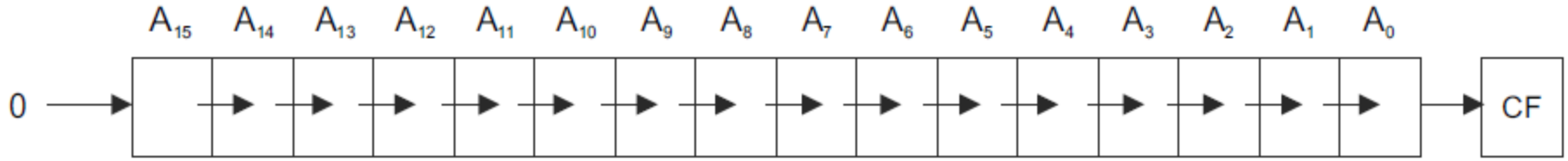
Instruction Set

5. Logical & Bit Manipulation Instructions

SHR (Shift logical right)

$A_n \ A_{n+1}$, CF←A_0, A_{15}←0, All flags are affected

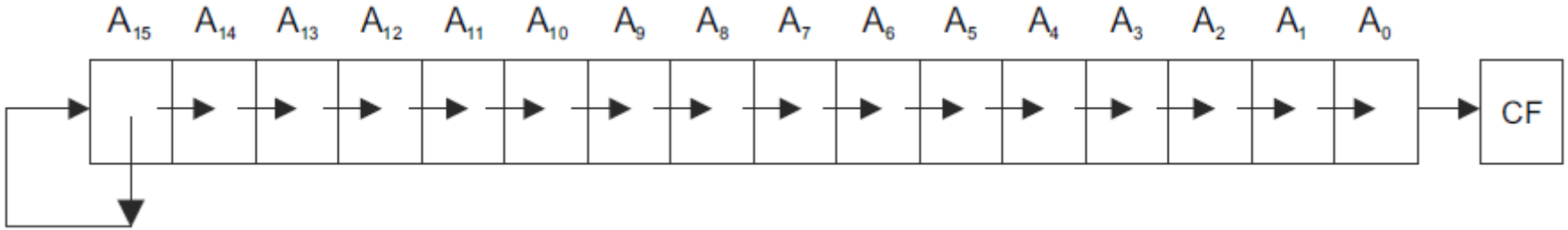
SHR AX, CL; SHR AX, 1



SAR (Shift arithmetic right)

A_n ←A_{n+1}, CF←A_0, A_{15} ← A_{15}, All flags are affected

SHR AX, CL; SAR AX,1



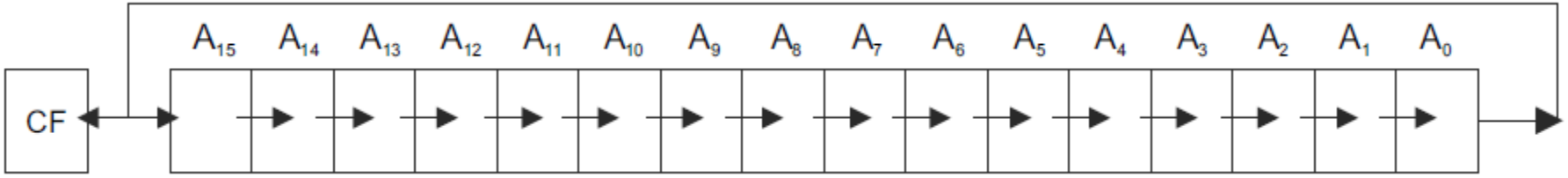
Instruction Set

5. Logical & Bit Manipulation Instructions

ROR(Rotate right without carry)

$A_n \leftarrow A_{n+1}$, $A_{15} \leftarrow A_0$, $CF \leftarrow A_0$, All flags are affected

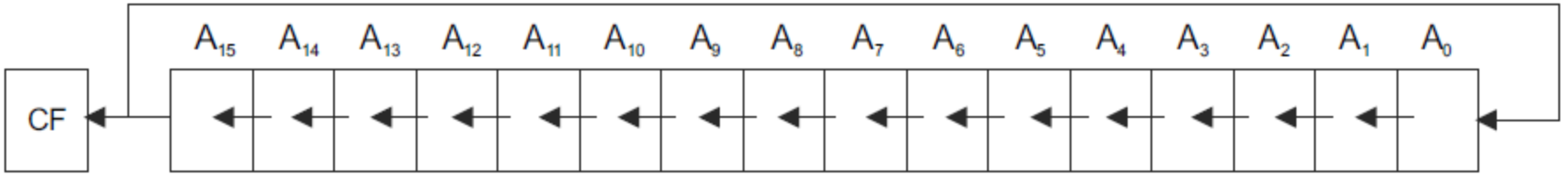
ROR AX, CL; ROR AX, 1



ROL (Rotate left without carry)

$A_{n+1} \leftarrow A_n$, $A_0 \leftarrow A_{15}$, $CF \leftarrow A_{15}$, All flags are affected

ROL AX, 1; ROL AX, CL



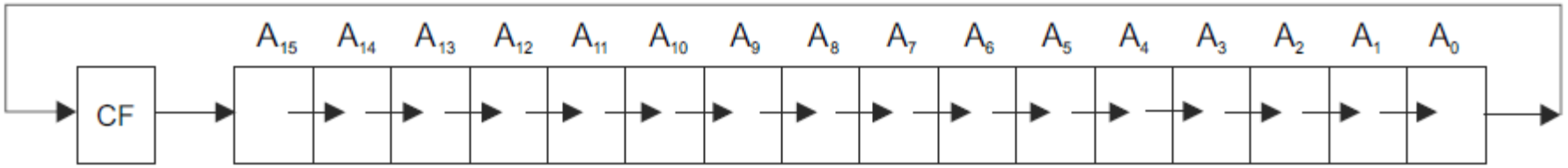
Instruction Set

5. Logical & Bit Manipulation Instructions

RCR (Rotate right through carry)

$A_n \leftarrow A_{n+1}, A_{15} \leftarrow CF, CF \leftarrow A_0$; All flags are affected

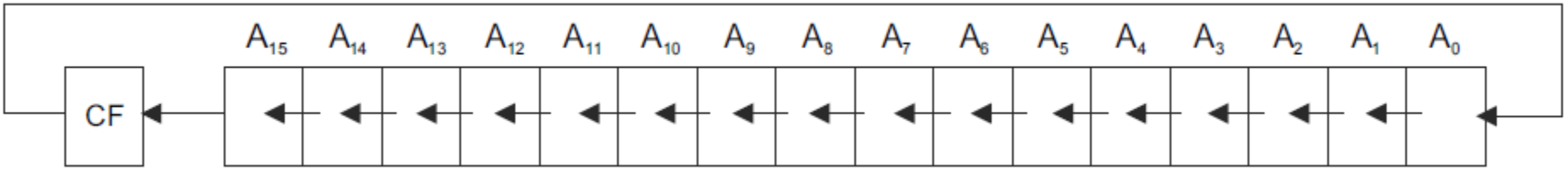
RCR AX, 1; RCR AX, CL



RCL (Rotate left Through Carry)

$A_{n+1} \leftarrow A_n, CF \leftarrow A_{15}, A_0 \leftarrow CF$ All flags are affected

RCL AX, 1; RCL AX, CL



Instruction Set

6. Jump Instructions

JMP target (Unconditional jump to target)

- Transfers the control of execution to the specified address using an 8-bit or 16-bit displacement or CS: IP.

Short : IP ← (IP + (target displacement sign-extended))
Near: IP ← (IP + (target displacement))
Indirect: IP ← (register or value in memory)
Far: CS ← targ_seg;
IP ← targ_offset, AX ← (AL * source 8)
Flag affected: None

JCXZ Target(short) (Jump short if CX register is 0)

Instruction Set

6. Jump Instructions

Jcond (Jump on condition)

Instruction	*Condition*	*Operation*
JO	O = 1	Jump on overflow set
JNO	O = 0	Jump on overflow clear
JB / JNAE	C = 1	Jump if below / Jump if not above or equal
JAE / JNB	C = 0	Jump if above or equal / Jump if not below
JE / JNZ	Z = 1	Jump if equal / Jump if not zero
JNE / JNZ	Z = 0	Jump if not equal / Jump if not zero
JBE / JNA	C = 1or Z = 1	Jump if below or equal / Jump if not above
JA / JNBE	C = 0 and Z = 0	Jump if above / Jump if not below or equal
JS	S = 1	Jump on sign set
JNS	S = 0	Jump on sign clear
JP / JPE	P = 1	Jump on parity bit set (parity even)
JNP / JPO	P = 0	Jump on parity bit clear (parity odd)
JL / JNGE	S = 1 or O = 1	Jump if less / Jump if not greater than or equal to
JGE / JNL	S = O	Jump if greater than or equal to / Jump if not less
JLE / JNG	Z = 1 or S and O = 1	Jump if less than or equal to / Jump if not greater than
JG / JNLE	Z = 0 or S = O	Jump if greater than / Jump if not less than or equal to

Instruction Set

7. Loop Instructions

- Executes the part of the program from the level or address specified in the instruction up to the loop instruction, CX number of times.
- After each iteration, CX is decremented automatically

```
        LEA SI, 0100;     Load SI with source address of data
        LEA DI, 0200;     Load DI with destination address of data
        MOV CX, 0009;     Number of bytes 9 is loaded in CX register
START   LODSB;            Data byte to AL and increment SI by 1
        STOSB;            The content of AL is stored in destination address represented by DI and and
                          increment DI by 1
        LOOP START;       repeat until CX = 0
```

Instruction Set

7. Loop Instructions

Instruction	Equivalent to:
LODSB	MOV AL, [SI] INC SI
STOSB	MOV [DI], AL INC DI
LOOP	DEC CX JNZ START

Instruction Set

7. Loop Instructions

LOOP Target(short) (Loop to short target) $CX \leftarrow (CX\text{-}1);$

Jump if CX ! = 0. The CX register is decremented by 1. If CX now is not equal to 0, the loop is back to target.

LOOPE/Z Target(short) (Loop to short target if Z bit set) $CX \leftarrow (CX\text{-}1);$

Jump if CX! = 0 and ZF = 1. The CX register is decremented by 1. If CX is not equal to 0 or if the Z bit is set, the loop is back to short target.

LOOPNE/NZ (Loop to short target if Z bit is clear) $CX \leftarrow (CX\text{-}1);$

Jump if CX! = 0 and ZF = 0. The CX register is decremented by 1. If CX is not equal to 0 or if the Z bit is clear, the loop is back to short target.

Instruction Set

8. Call & Return Instructions

CALL target (Call a procedure) Near call: PUSH IP, JMP to target

SP ← IP, SP ← SP-2, IP← IP + DISP

Far call: PUSH CS, PUSH IP, JMP to target :Flag affected: None

SP ←CS, SP ← SP-2, SP ← IP, SP ← SP-2, IP← 16 bit DATA

CS← 16-bit DATA

RET (Return from procedure)

RET n (return from procedure and add n to SP)

Near return: POP IP

IP ← SP, SP ← SP + 2

The syntax for a near return is RET.

Far return: POP IP, POP CS; Flag affected: None

IP ← SP, SP ← SP + 2, CS← SP, SP ← SP + DISP

The syntax for a far return is RET FAR.

Instruction Set

8. Call & Return Instructions

INT n (Interrupt (software))

PUSHF; IF ← 0; TF ← 0; PUSH CS; PUSH IP

IP ← 0000: [type * 4];

CS ← 0000: [(type * 4) + 2]; Flag affected: IT

INTO (Interrupt on overflow) If OF = 1, then perform INT through vector 4; Flag affected: None

IRET (Return from interrupt service routine)

POP IP; POP CS; POPF AX (AL * src8); Flag affected: All

Instruction Set

9. String Instructions

MOVSB (Move string byte) ES: [DI] ← DS: [SI]; DI = DI ± 1; SI = SI ± 1; Flag affected: None.

MOVSW (Move string word) ES: [DI] ← DS: [SI]; DI = DI ± 2; SI = SI ± 2; Flag affected: None

STOSB (Store string byte) ES: [DI] ← AL; DI = DI ± 1; Flag affected: None

STOSW (Store string word) ES: [DI] ← AX; DI = DI±2; Flag affected: None

LODSB (Load string byte) AL ← DS: [SI]; SI = SI±1; Flag affected: None

LODSW (Load string word) AX ← DS: [SI]; SI = SI±2; Flag affected: None

Instruction Set

9. String Instructions

CMPSB (Compare string byte) Flags←(result of CMP DS: [SI], ES: [DI]); DI = DI±1; SI = SI±1; Flag affected: as CMP

CMPSW (Compare string word) Flags ← (result of CMP DS: [SI], ES: [DI]); DI = DI ± 2; SI = SI ± 2; Flag affected: as CMP

SCASB (Scan string byte) Flags ← (result of CMP ES: [DI], AL); DI = DI ± 1; Flag affected: same as CMP instruction

SCASW (Scan string word) Flags ← (result of CMP ES: [DI], AX); DI = DI ± 2; Flag affected: same as CMP instruction

Instruction Set

9. String Instructions

REPEAT Instructions

The common REP instructions are REP (repeat), REPE (repeat while equal), REPZ (repeat while zero), REPNE (repeat while not equal), and REPNZ (repeat while not zero)

REP / REPE / REPZ (Repeat string instruction (prefix))

CX ← (CX-1); until CX = 0;

Flag affected: Z

This is a prefix byte that forces a string operation to be repeated as long as CX is not equal to 0. CX is decremented once for each repetition. The object code of REPZ/REPE instruction is 1111 001Z = F3 as Z = 1.

REPNE / REPNZ (Repeat string instruction while not zero (prefix))

ZF← 0; CX ← (CX-1); String Operation repeats while (CX! = 0 and ZF! = 0); Flag affected: Z. This is a prefix byte that keeps a string operation repeating while CX is not zero and Z! = 0. The object code of REPNZ/REPNE instruction is 1111 001Z = F2 as Z = 0.

Instruction Set

10. Processor Control Instructions

- Type 1 - Flag Manipulation Instructions

CLC (Clear the carry flag)	CF ← 0; Flag affected: C.
CMC (Complement the carry flag)	CF ← ~CF; Flag affected: C
STC (Set the carry flag)	CF ← 1; Flag affected: C
CLD (Clear direction flag)	DF ← 0; Flag affected: D
STD (Set direction flag)	DF ← 1; Flag affected: D
CLI (Clear interrupt flag)	IF ← 0; Flag affected: IF
STI (Set interrupt flag)	IF ← 1; Flag affected: IF

Instruction Set

10. Processor Control Instructions

- Type 2 – Machine Control Instructions

HLT (Halt)

WAIT (Wait)

LOCK (Lock bus)

NOP (No operation)

TEST

ESC (Escape)