ECE 424 Design of Microprocessor-Based Systems

80x86 Instructions

Part 1

Haibo Wang
ECE Department
Southern Illinois University
Carbondale, IL 62901

Instruction Types

- Data transfer instructions
- String instructions
- Arithmetic instructions
- Bit manipulation instructions
- Loop and jump instructions
- Subroutine and interrupt instructions
- Processor control instructions

An excellent website about 80x86 instruction set: *http://www.penguin.cz/~literakl/intel/intel.html*
Another good reference is in the tutorial of 8086 emulator

Addressing Modes

Addressing Modes	*Examples*
❑ Immediate addressing	MOV AL, 12H
❑ Register addressing	MOV AL, BL
❑ Direct addressing	MOV [500H], AL
❑ Register Indirect addressing	MOV DL, [SI]
❑ Based addressing	MOV AX, [BX+4]
❑ Indexed addressing	MOV [DI-8], BL
❑ Based indexed addressing	MOV [BP+SI], AH
❑ Based indexed with displacement addressing	MOV CL, [BX+DI+2]

Exceptions

❑ String addressing

❑ Port addressing (*e.g.* IN AL, 79H)

Flag Register

- ❑ Flag register contains information reflecting the current status of a microprocessor. It also contains information which controls the operation of the microprocessor.

15														0
—	NT	IOPL	OF	DF	IF	TF	SF	ZF	—	AF	—	PF	—	CF

➢ Control Flags

IF:	Interrupt enable flag
DF:	Direction flag
TF:	Trap flag

➢ Status Flags

CF:	Carry flag
PF:	Parity flag
AF:	Auxiliary carry flag
ZF:	Zero flag
SF:	Sign flag
OF:	Overflow flag
NT:	Nested task flag
IOPL:	Input/output privilege level

Flags Commonly Tested During the Execution of Instructions

- There are five flag bits that are commonly tested during the execution of instructions

  - Sign Flag (Bit 7), SF: 0 for positive number and 1 for negative number

  - Zero Flag (Bit 6), ZF: If the ALU output is 0, this bit is set (1); otherwise, it is 0

  - Carry Flag (Bit 0), CF: It contains the carry generated during the execution

  - Auxiliary Carry, AF: (Bit 4) Depending on the width of ALU inputs, this flag bit contains the carry generated at bit 3 (or, 7, 15) of the 8088 ALU

  - Parity Flag (bit2), PF: It is set (1) if the output of the ALU has even number of ones; otherwise it is zero

Data Transfer Instructions

- ❑ MOV *Destination, Source*
  - — Move data from source to destination; *e.g.* ***MOV [DI+100H], AH***
  - — It does not modify flags

➢ For 80x86 family, directly moving data from one memory location to another memory location is not allowed

MOV [SI], [5000H] ✗

➢ When the size of data is not clear, assembler directives are used

MOV [SI], 0 ?

▪ ***BYTE PTR***	MOV BYTE PTR [SI], 12H
▪ ***WORD PTR***	MOV WORD PTR [SI], 12H
▪ ***DWORD PTR***	MOV DWORD PTR [SI], 12H

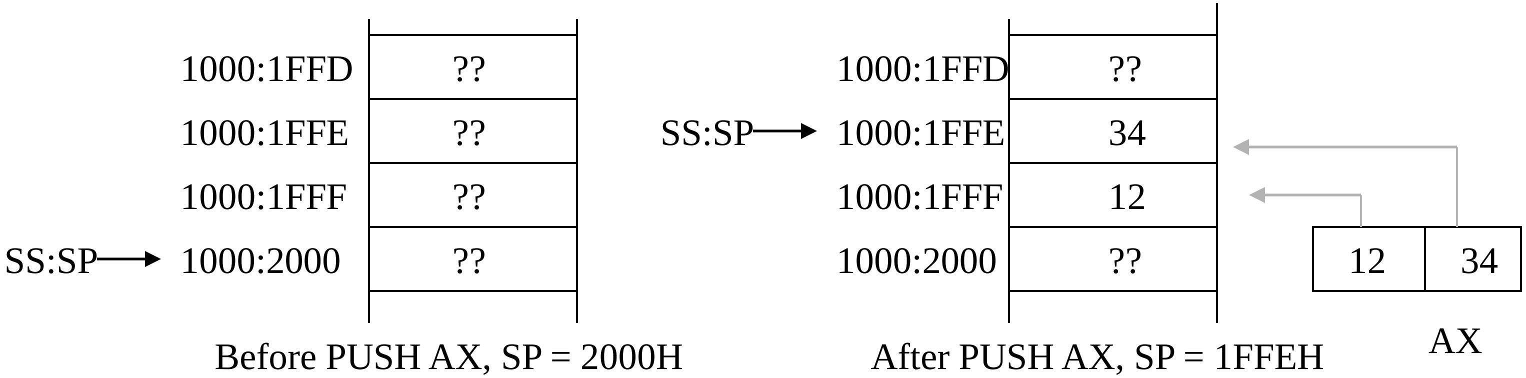
➢ You can not move an immediate data to segment register by MOV

MOV DS, 1234H ✗

Instructions for Stack Operations

- ❑ What is a Stack ?
  - — A stack is a collection of memory locations. It always follows the rule of last-in-firs-out
  - — Generally, SS and SP are used to trace where is the latest date written into stack

- ❑ PUSH *Source*
  - — Push data (***word***) onto stack
  - — It does not modify flags
  - — For Example: PUSH AX (assume ax=1234H, SS=1000H, SP=2000H before PUSH AX)



	1000:1FFD	??
	1000:1FFE	??
	1000:1FFF	??
SS:SP →	1000:2000	??

Before PUSH AX, SP = 2000H

	1000:1FFD	??
SS:SP →	1000:1FFE	34
	1000:1FFF	12
	1000:2000	??

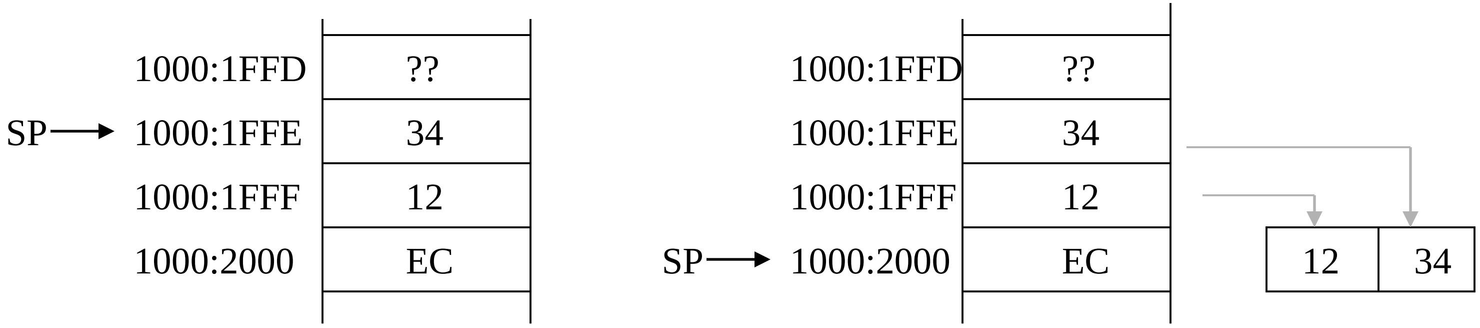
After PUSH AX, SP = 1FFEH

➢ Decrementing the stack pointer during a push is a standard way of implementing stacks in hardware

Instructions for Stack Operations

- ❑ PUSHF
  - — Push the values of the flag register onto stack
  - — It does not modify flags

- ❑ POP *Destination*
  - — Pop word off stack
  - — It does not modify flags
  - — For example: ***POP AX***



	Address	Content
	1000:1FFD	??
SP →	1000:1FFE	34
	1000:1FFF	12
	1000:2000	EC

Before POP, SP = 1FFEH

	Address	Content
	1000:1FFD	??
	1000:1FFE	34
	1000:1FFF	12
SP →	1000:2000	EC

12	34

AX

After POP AX, SP = 2000H

- ❑ POPF
  - — Pop word from the stack to the flag register
  - — It modifies all flags

Data Transfer Instructions

- ❑ SAHF
  - — Store data in AH to the low 8 bits of the flag register
  - — It modifies flags: AF, CF, PF, SF, ZF
- ❑ LAHF
  - — Copies bits 0-7 of the flags register into AH
  - — It does not modify flags
- ❑ LDS *Destination Source*
  - — Load 4-byte data (pointer) in memory to two 16-bit registers
  - — Source operand gives the memory location
  - — The first two bytes are copied to the register specified in the destination operand; the second two bytes are copied to register DS
  - — It does not modify flags
- ❑ LES *Destination Source*
  - — It is identical to LDS except that the second two bytes are copied to ES
  - — It does not modify flags

Data Transfer Instructions

- LEA *Destination Source*
  - — Transfers the offset address of source (must be a memory location) to the destination register
  - — It does not modify flags

- XCHG *Destination Source*
  - — It exchanges the content of destination and source
  - — One operand must be a microprocessor register, the other one can be a register or a memory location
  - — It does not modify flags

- XLAT
  - — Replace the data in AL with a data in a user defined look-up table
  - — BX stores the beginning address of the table
  - — At the beginning of the execution, the number in AL is used as the index of the look-up table
  - — It does not modify flags

String Instructions

- ❑ String is a collection of bytes, words, or long-words that can be up to 64KB in length
- ❑ String instructions can have at most two operands. One is referred to as source string and the other one is called destination string
  - — Source string must locate in Data Segment and SI register points to the current element of the source string
  - — Destination string must locate in Extra Segment and DI register points to the current element of the destination string

DS : SI		
0510:0000	53	S
0510:0001	48	H
0510:0002	4F	O
0510:0003	50	P
0510:0004	50	P
0510:0005	45	E
0510:0006	52	R

Source String

ES : DI		
02A8:2000	53	S
02A8:2001	48	H
02A8:2002	4F	O
02A8:2003	50	P
02A8:2004	50	P
02A8:2005	49	I
02A8:2006	4E	N

Destination String

Repeat Prefix Instructions

- REP *String Instruction*

  — The prefix instruction makes the microprocessor repeatedly execute the string instruction until CX decrements to 0 (During the execution, CX is decreased by one when the string instruction is executed one time).

  — For Example:

```
MOV CX, 5
REP MOVSB
```

By the above two instructions, the microprocessor will execute MOVSB 5 times.

— Execution flow of REP MOVSB::

```
While (CX!=0)
 {
   CX = CX –1;
   MOVSB;
 }
```

OR

```
Check_CX:  If CX!=0 Then
                 CX = CX –1;
                 MOVSB;
                 goto Check_CX;
             end if
```

Repeat Prefix Instructions

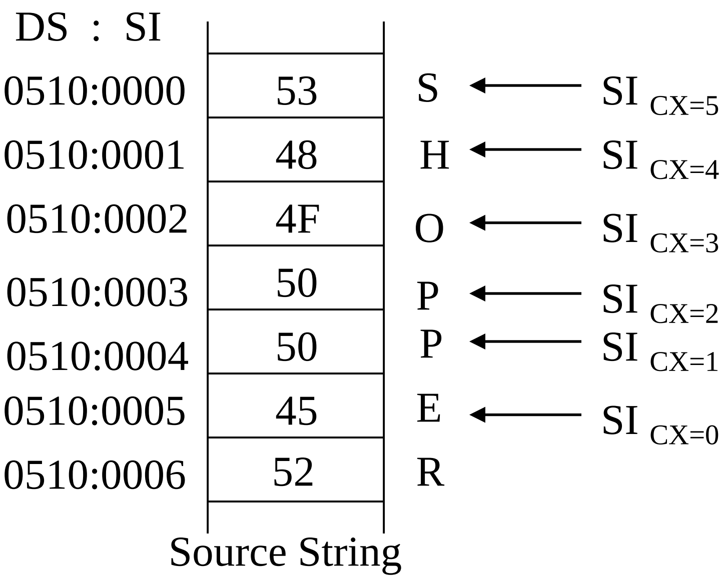
- ❑ REPZ *String Instruction*
  - — Repeat the execution of the string instruction until CX=0 or zero flag is clear
- ❑ REPNZ *String Instruction*
  - — Repeat the execution of the string instruction until CX=0 or zero flag is set
- ❑ REPE *String Instruction*
  - — Repeat the execution of the string instruction until CX=0 or zero flag is clear
- ❑ REPNE *String Instruction*
  - — Repeat the execution of the string instruction until CX=0 or zero flag is set

Direction Flag

- ❑ Direction Flag (DF) is used to control the way SI and DI are adjusted during the execution of a string instruction
  - — DF=0, SI and DI will auto-increment during the execution; otherwise, SI and DI auto-decrement
  - — Instruction to set DF: ***STD***; Instruction to clear DF: ***CLD***
  - — Example:

```
CLD
MOV CX, 5
REP MOVSB
```

At the beginning of execution,
DS=0510H and SI=0000H



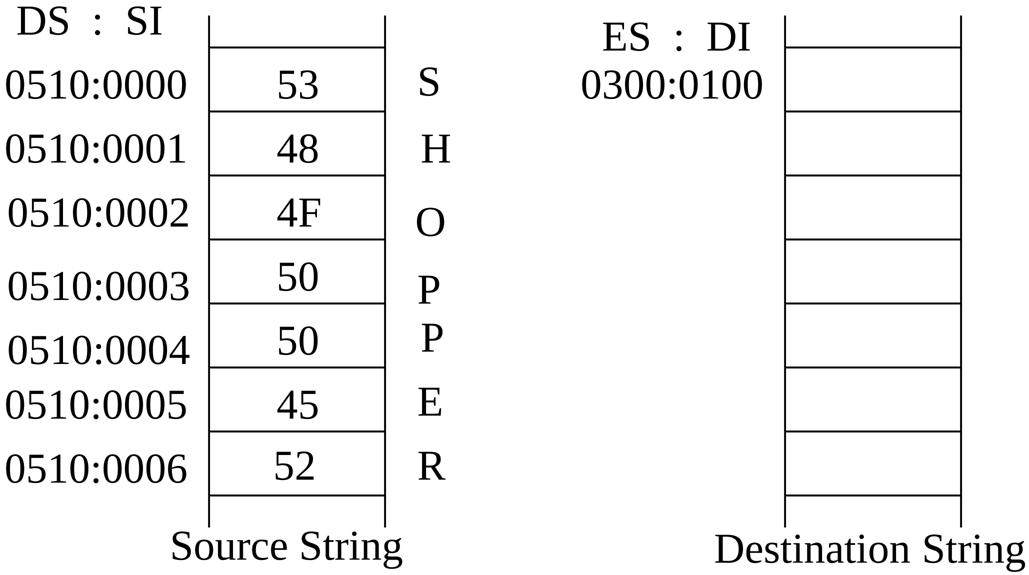
DS : SI			
0510:0000	53	S	← SI $_{CX=5}$
0510:0001	48	H	← SI $_{CX=4}$
0510:0002	4F	O	← SI $_{CX=3}$
0510:0003	50	P	← SI $_{CX=2}$
0510:0004	50	P	← SI $_{CX=1}$
0510:0005	45	E	← SI $_{CX=0}$
0510:0006	52	R	

Source String

String Instructions

- ❑ MOVSB (MOVSW)
  - — Move byte (word) at memory location DS:SI to memory location ES:DI and update SI and DI according to DF and the width of the data being transferred
  - — It does not modify flags
  - —Example:

```
MOV  AX, 0510H
MOV DS, AX
MOV SI, 0
MOV AX, 0300H
MOV ES, AX
MOV DI, 100H
CLD
MOV CX, 5
REP MOVSB
```



DS : SI		
0510:0000	53	S
0510:0001	48	H
0510:0002	4F	O
0510:0003	50	P
0510:0004	50	P
0510:0005	45	E
0510:0006	52	R

Source String

ES : DI
0300:0100

Destination String

String Instructions

- ❑ CMPSB (CMPSW)

— Compare bytes (words) at memory locations DS:SI and ES:DI;
update SI and DI according to DF and the width of the data being compared

— It modifies flags

—Example:

Assume: ES = 02A8H
DI = 2000H
DS = 0510H
SI = 0000H

```
CLD
MOV CX, 9
REPZ CMPSB
```

What's the values of CX after
The execution?

DS : SI		
0510:0000	53	S
0510:0001	48	H
0510:0002	4F	O
0510:0003	50	P
0510:0004	50	P
0510:0005	45	E
0510:0006	52	R

Source String

ES : DI		
02A8:2000	53	S
02A8:2001	48	H
02A8:2002	4F	O
02A8:2003	50	P
02A8:2004	50	P
02A8:2005	49	I
02A8:2006	4E	N

Destination String

String Instructions

- ❑ SCASB (SCASW)

  — Move byte (word) in AL (AX) and at memory location ES:DI;
  update DI according to DF and the width of the data being compared

  — It modifies flags

- ❑ LODSB (LODSW)

  — Load byte (word) at memory location DS:SI to AL (AX);
  update SI according to DF and the width of the data being transferred

  — It does not modify flags

- ❑ STOSB (STOSW)

  — Store byte (word) at in AL (AX) to memory location ES:DI;
  update DI according to DF and the width of the data being transferred

  — It does not modify flags