Microprocessors & Interfacing

INSTRUCTION SET

BITS Pilani

Dr. Gargi Prabhu
Department of CS & IS

Register Subtraction

- Subtracts the 16-bit contents of registers CX and DX from the contents of register BX

E.g. SUB BX,CX

SUB BX,DX

Immediate Subtraction

- Microprocessor also allows immediate operands for the subtraction of constant data

E.g.

MOV CH,22H

SUB CH,44H

Z = (result not zero)
C = (borrow)
A = (half-borrow)
S = (result negative)
P = (even parity)
O = (no overflow)

Immediate Subtraction

- Microprocessor also allows immediate operands for the subtraction of constant data

E.g.

MOV CH,22H

SUB CH,44H

$Z = 0$ (result not zero)

$C = 1$ (borrow)

$A = 1$ (half-borrow)

$S = 1$ (result negative)

$P = 1$ (even parity)

$O = 0$ (no overflow)

Example

```
0859:0102 80ED44            SUB     CH,44
-
AX=0000 BX=FFFF CX=DE00 DX=FFFE SP=FFFC BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0105 NV UP EI NG NZ AC PE CY
0859:0105 FFF7              PUSH    DI
-_
```

$Z = 0$ (result not zero)

$C = 1$ (borrow)

$A = 1$ (half-borrow)

$S = 1$ (result negative)

$P = 1$ (even parity)

$O = 0$ (no overflow)

Flag Name	Set	Clear
Overflow(yes/no)	OV	NV
Direction(increment/ decrement)	DN	UP
Interrupt(enable/disable)	EI	DI
Sign(negative/positive)	NG	PL
Zero(yes/no)	ZR	NZ
Auxiliary carry(yes/no)	AC	NA
Parity(even/odd)	PE	PO
Carry(yes/no)	CY	NC

Assembly Language	*Operation*
SUB CL,BL	CL = CL – BL
SUB AX,SP	AX = AX – SP
SUB ECX,EBP	ECX = ECX – EBP
SUB RDX,R8	RDX = RDX – R8 (64-bit mode)
SUB DH,6FH	DH = DH – 6FH
SUB AX,0CCCCH	AX = AX – 0CCCCH
SUB ESI,2000300H	ESI = ESI – 2000300H
SUB [DI],CH	Subtracts CH from the byte contents of the data segment memory addressed by DI and stores the difference in the same memory location
SUB CH,[BP]	Subtracts the byte contents of the stack segment memory location addressed by BP from CH and stores the difference in CH
SUB AH,TEMP	Subtracts the byte contents of memory location TEMP from AH and stores the difference in AH
SUB DI,TEMP[ESI]	Subtracts the word contents of the data segment memory location addressed by TEMP plus ESI from DI and stores the difference in DI
SUB ECX,DATA1	Subtracts the doubleword contents of memory location DATA1 from ECX and stores the difference in ECX
SUB RCX,16	RCX = RCX – 18 (64-bit mode)

DEC Instruction

- Decrement subtraction (DEC) subtracts 1 from a register or the contents of a memory location

E.g.

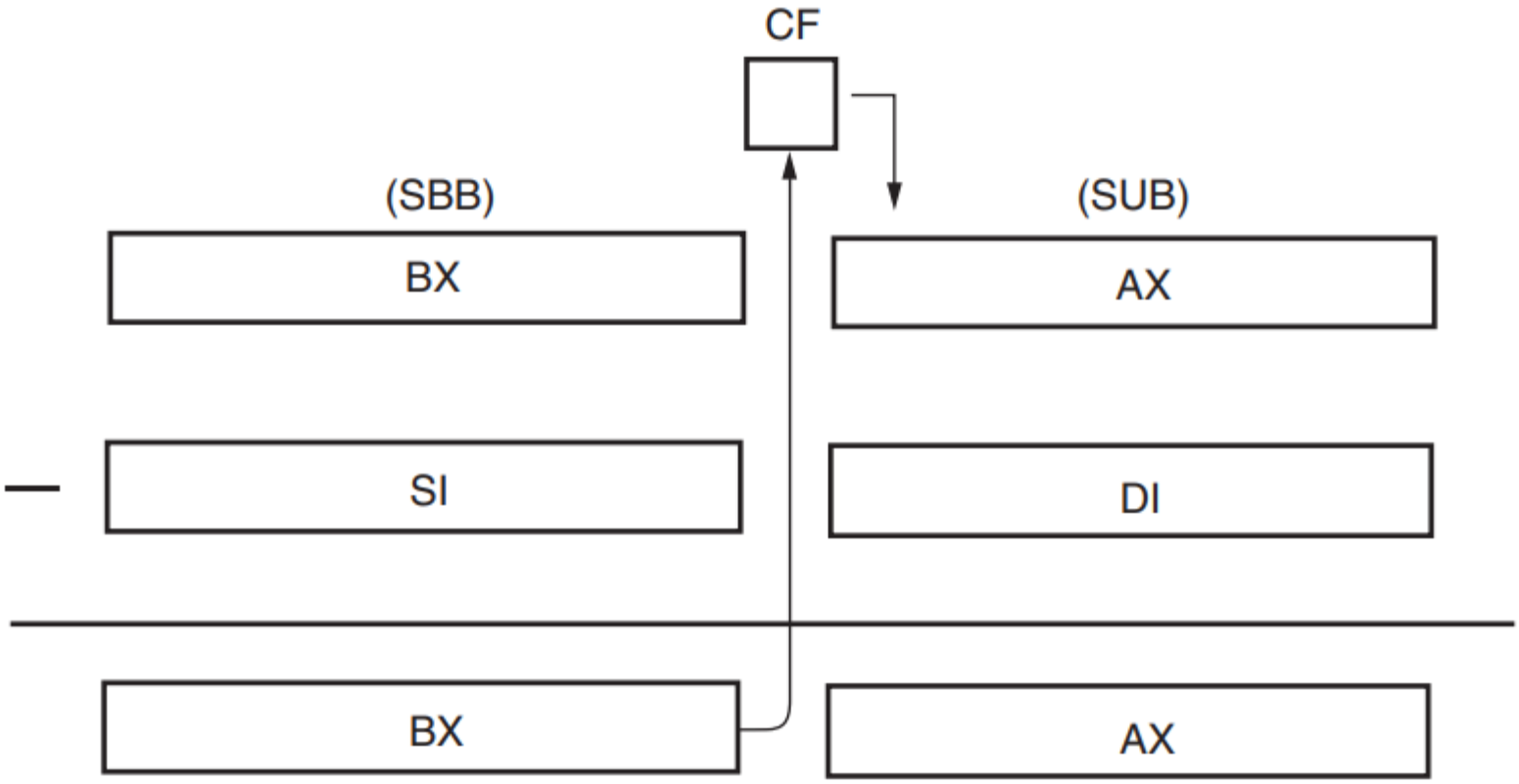
DEC BH	BH = BH – 1
DEC CX	CX = CX – 1

DEC Example

Assembly Language	*Operation*
DEC BH	BH = BH – 1
DEC CX	CX = CX – 1
DEC EDX	EDX = EDX – 1
DEC R14	R14 = R14 – 1 (64-bit mode)
DEC BYTE PTR[DI]	Subtracts 1 from the byte contents of the data segment memory location addressed by DI
DEC WORD PTR[BP]	Subtracts 1 from the word contents of the stack segment memory location addressed by BP
DEC DWORD PTR[EBX]	Subtracts 1 from the doubleword contents of the data segment memory location addressed by EBX
DEC QWORD PTR[RSI]	Subtracts 1 from the quadword contents of the memory location addressed by RSI (64-bit mode)
DEC NUMB	Subtracts 1 from the contents of data segment memory location NUMB

Subtraction-with-Borrow

SUB AX,DI
SBB BX,SI



Multiplication

- Multiplication is performed on bytes, words, or doublewords, and can be signed integer (IMUL) or unsigned integer (MUL)
- The product after a multiplication is always a double-width product.
- Two 8-bit numbers ->16 bit
- Two 16-bit numbers -> 32 bit
- Two 32-bit numbers -> 64 bit
- Flag bits (overflow and carry) change when the multiply instruction executes and produce predictable outcomes
- In 8-bit multiplication, if the most significant 8 bits of the result are zero, both C and O flag bits equal zero.

Multiplication

- These flag bits show that the result is 8 bits wide (C=0) or 16 bits wide (C=1).
- In a 16-bit multiplication, if the most significant 16-bits part of the product is 0, both C and O clear to zero.
- In a 32-bit multiplication, both C and O indicate that the most significant 32 bits of the product are zero.

8-Bit Multiplication

- With 8-bit multiplication, the multiplicand is always in the AL register, whether signed or unsigned.
- The multiplier can be any 8-bit register or any memory location.
- Immediate multiplication is not allowed.

Assembly Language	*Operation*
MUL CL	AL is multiplied by CL; the unsigned product is in AX
IMUL DH	AL is multiplied by DH; the signed product is in AX
IMUL BYTE PTR[BX]	AL is multiplied by the byte contents of the data segment memory location addressed by BX; the signed product is in AX
MUL TEMP	AL is multiplied by the byte contents of data segment memory location TEMP; the unsigned product is in AX

Can we replace AL?

DX=BX × CL

```
MOV   BL,5
MOV   CL,10
MOV   AL,CL
MUL   BL
MOV   DX,AX
```

Can we replace AL?

DX=BX × CL

DosBox Example

```
D:\MPSOFT~1\DEBUG125>debugx
-a 100
0859:0100 mov bx,5
0859:0103 mov cx,2
0859:0106 mul bx,cx
                  ^ Error
0859:0106 mov ax,2
0859:0109 mul bx
0859:010B
```

DosBox Example

```
D:\MPSOFT~1\DEBUG125>debugx
-a 100
0859:0100 mov b
0859:0103 mov c
0859:0106 mul b

0859:0106 mov a
0859:0109 mul b
0859:010B
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
0859:0106 mov ax,2
0859:0109 mul bx
0859:010B mov dx,ax
0859:010D
-t=100
AX=0000 BX=0005 CX=0000 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0103 NV UP EI NG NZ NA PE NC
0859:0103 B90200            MOV     CX,0002
-
AX=0000 BX=0005 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0106 NV UP EI NG NZ NA PE NC
0859:0106 B80200            MOV     AX,0002
-
AX=0002 BX=0005 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=0109 NV UP EI NG NZ NA PE NC
0859:0109 F7E3              MUL     BX
-
AX=000A BX=0005 CX=0002 DX=0000 SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=010B NV UP EI NG NZ NA PE NC
0859:010B 89C2              MOV     DX,AX
-
AX=000A BX=0005 CX=0002 DX=000A SP=FFFE BP=0000 SI=0000 DI=0000
DS=0859 ES=0859 SS=0859 CS=0859 IP=010D NV UP EI NG NZ NA PE NC
0859:010D 0000              ADD     [BX+SI],AL                  DS:0005=EA
-
```

16-bit Multiplication Example

- AX contains the multiplicand instead of AL
- 32-bit product appears in DX–AX instead of AX
- The DX register always contains the most significant 16 bits of the product, and AX contains the least significant 16 bits.

Assembly Language	*Operation*
MUL CX	AX is multiplied by CX; the unsigned product is in DX–AX
IMUL DI	AX is multiplied by DI; the signed product is in DX–AX
MUL WORD PTR[SI]	AX is multiplied by the word contents of the data segment memory location addressed by SI; the unsigned product is in DX–AX

Division

- 8-bit or 16-bit numbers in the 8086–80286 microprocessors, and on 32-bit numbers in the 80386 and above microprocessor
- signed (IDIV) or unsigned (DIV) integers.
- Dividend is always a double-width dividend that is divided by the operand.
- 8-bit division divides a 16-bit number by an 8-bit number
- 16-bit division divides a 32-bit number by a 16-bit number
- 32-bit division divides a 64-bit number by a 32-bit number.
- There is no immediate division instruction available to any microprocessor.

Division

- None of the flag bits change predictably for a division.
- A division can result in two different types of errors; one is an attempt to divide by zero and the other is a divide overflow.
- A divide overflow occurs when a small number divides into a large number.
- E.g. AX=3000 divided by 2

AL=1500 -> Overflow->Interrupt

8-Bit Division

- 8-bit division uses the AX register to store the dividend that is divided by the contents of any 8-bit register or memory location
- Quotient moves into AL after the division with AH containing a whole number remainder.
- For a signed division, the quotient is positive or negative; the remainder always assumes the sign of the dividend and is always an integer.

E.g. AX=0010H (+16)

BL=0FDH (-3)

IDIV BL

AX= 01FBH

What about -16 divided by +3?

8-bit Division

innovate achieve lead

Assembly Language	*Operation*
DIV CL	AX is divided by CL; the unsigned quotient is in AL and the unsigned remainder is in AH
IDIV BL	AX is divided by BL; the signed quotient is in AL and the signed remainder is in AH
DIV BYTE PTR[BP]	AX is divided by the byte contents of the stack segment memory location addressed by BP; the unsigned quotient is in AL and the unsigned remainder is in AH

```
MOV   AL,NUMB        ;get NUMB
MOV   AH,0           ;zero-extend
DIV   NUMB1          ;divide by NUMB1
MOV   ANSQ,AL        ;save quotient
MOV   ANSR,AH        ;save remainder
```

16-Bit Division

- Sixteen-bit division is similar to 8-bit division, except that instead of dividing into AX, 16-bit number is divided into DX–AX, a 32-bit dividend
- If a 16-bit unsigned number is placed in AX, DX must be cleared to zero.

```
MOV   AX,-100          ;load a -100
MOV   CX,9             ;load +9
CWD                    ;sign-extend
IDIV   CX
```

16-Bit Division

Assembly Language	*Operation*
DIV CX	DX–AX is divided by CX; the unsigned quotient is AX and the unsigned remainder is in DX
IDIV SI	DX–AX is divided by SI; the signed quotient is in AX and the signed remainder is in DX
DIV NUMB	DX–AX is divided by the word contents of data segment memory NUMB; the unsigned quotient is in AX and the unsigned remainder is in DX

Basic Logic Instructions

- AND, OR, Exclusive-OR, and NOT
- Logic operations provide binary bit control in low-level software. The logic instructions allow bits to be set, cleared, or complemented.
- When binary data are manipulated in a register or a memory location, the rightmost bit position is always numbered bit 0.
- Bit position numbers increase from bit 0 toward the left, to bit 7 for a byte, and to bit 15 for a word.
- A doubleword (32 bits) uses bit position 31 as its leftmost bit and a quadword (64-bits) uses bit position 63 as it leftmost bit.

AND

- The AND operation performs logical multiplication
- 0 AND anything is always 0, and 1 AND 1 is always 1.

A	B	T
0	0	0
0	1	0
1	0	0
1	1	1

AND

Assembly Language	*Operation*
AND AL,BL	AL = AL and BL
AND CX,DX	CX = CX and DX
AND ECX,EDI	ECX = ECX and EDI
AND RDX,RBP	RDX = RDX and RBP (64-bit mode)
AND CL,33H	CL = CL and 33H
AND DI,4FFFH	DI = DI and 4FFFH
AND ESI,34H	ESI = ESI and 34H
AND RAX,1	RAX = RAX and 1 (64-bit mode)
AND AX,[DI]	The word contents of the data segment memory location addressed by DI are ANDed with AX
AND ARRAY[SI],AL	The byte contents of the data segment memory location addressed by ARRAY plus SI are ANDed with AL
AND [EAX],CL	CL is ANDed with the byte contents of the data segment memory location addressed by ECX

Thank You