CSE-3103: Microprocessor and Microcontroller

Dept. of Computer Science and Engineering
University of Dhaka

Prof. Sazzad M.S. Imran, PhD
Dept. of Electrical and Electronic Engineering
sazzadmsi.webnode.com

Arithmetic Instructions

DIV/IDIV →

DIV = Divide → used for unsigned data,
IDIV = Integer Divide → used for signed data.

Syntax →

DIV/IDIV   divisor

AX/DX:AX ← dividend.
Can work with 8-bit or 16-bit operands.
Affects all six status flags →
[OF, SF, ZF, AF, PF, CF].

Example (divides 8 with 2) →

```
mov     ax, 80
mov     bl, 02
div     bl
; al ← ax/bl
; ah ← ax%bl
```

Two cases of division with different operand size →

(i) divisor is 1 byte:
AX ← dividend (16 bits)
AL ← quotient (8 bits)
AH ← remainder (8 bits)

(ii) divisor is 1 word:
DX:AX ← dividend (32 bits)
AX ← quotient (16 bits)
DX ← remainder (16 bits)

<u>Arithmetic Instructions</u>

ADC →

ADC = Add with Carry.

Destination operand ← destination operand + source operand + CF.

ADC AX, DX [AX ← AX + DX + CF]

Destination operand →

register or memory location.

Source operand →

immediate value, register or memory location.

Two memory operands cannot be used in one instruction.

CF = carry from previous addition.

Immediate value →

sign-extended to length of destination operand format.

Does not distinguish between signed or unsigned operands.

Processor evaluates result for both data types and

OF = 1 → carry in signed result,

CF = 1 → carry in unsigned result.

SF = sign of signed result.

Arithmetic Instructions

AAA →

ASCII Adjust after Addition.
Adjusts sum of two unpacked BCD values = unpacked BCD result.
AL = source and destination operand.

AAA instruction is only useful when it follows
AH = 00H before addition,
ADD two unpacked BCD values, and
AL register ← byte result.

AAA adjusts contents of AL register.
AL ← correct 1-digit unpacked BCD result.

After addition operation →
Lower nibble of AL = 0 to 9 and AF = 0 →
upper nibble of AL = 0H,
no change in lower nibble.

Lower nibble of AL > 9 or AF =1 →
6 is added to lower nibble in AL,
upper nibble of AL = 0,
AH is incremented by 1,
AF and CF = 1.

Precise ASCII codes of sum = AX + 3030H.

Arithmetic Instructions

AAA →

1) AL = 57 ; before to AAA
AL = 07 ; after AAA execution

2) AL = 5A ; previous to AAA
AH = 00
A > 9, hence A+6 = 1010 + 0110 = 10000B = 10H
AX = 005A ; previous to AAA
AX = 0100 ; after AAA execution

3)
```
sub   AH, AH        ; clear AH
mov   AL, '6'       ; AL := 36H
add   AL, '7'       ; AL := 36H + 37H = 6DH
aaa                 ; D > 9 → D+6 = 1101 + 0110 = 10011
                    ; AX := 0103H
or    AX, 3030H     ; AX := 3133H
```

Arithmetic Instructions

DAA →

Decimal Adjust Accumulator.
Addition of 2 packed BCD numbers = valid BCD number.
AL ← valid BCD number.

Lower nibble of AL > 9, or AF = 1 →
AL ← AL + 06H.
upper nibble of AL > 9 or CF = 1 →
AL ← AL + 60H.

Example →

CL = 29
i) AL = 53

```
ADD     AL, CL          ; AL ← (AL) + (CL) = 53 + 29 = 7C
DAA                     ; AL ← 7C + 06 = 82 (as C > 9)
```

ii) AL = 73

```
ADD     AL, CL          ; AL ← (AL) + (CL) = 73 + 29 = 9C
DAA                     ; 9C+06 = A2, AL ← A2 + 60 = 02 and CF=1
```

Arithmetic Instructions

SBB →

SBB = SuBtraction with Borrow
Destination operand ← destination operand – source operand – CF.
SBB AX, DX [AX ← AX – DX – CF]

Destination operand = register or memory location;
Source operand = immediate value, register, or memory location.
Two memory operands cannot be used in one instruction.
CF = borrow from previous subtraction.

Immediate value is sign-extended to length of destination operand.
Does not distinguish between signed or unsigned operands.

Processor evaluates result for both data types and

OF = 1 → borrow in signed result,
CF = 1 → borrow in unsigned result,
SF = sign of signed result.

Arithmetic Instructions

NEG →

Replaces value of destination operand with its two's complement.
Equivalent to subtracting operand from 0.
Destination operand = general-purpose register or memory location.

AAS →

ASCII Adjust after Subtraction.
Adjust subtraction of 2 unpacked BCD values = unpacked BCD result.
AL = source and destination operand.

AAS is only useful when it follows

SUB that subtracts 2 unpacked ASCII operands and
AL ← byte result.

Adjusts contents of AL.
AL ← correct 1-digit unpacked BCD result in decimal format.

Arithmetic Instructions

AAS →

Lower 4 bits of AL > 9 or AF = 1 →

AL is decremented by 6,
AH is decremented by 1,
CF and AF = 1.

Lower 4 bits of AL < 9 and AF = 0 →

CF and AF = 0,
result require no correction.

Upper nibble of AL = 0.

After adjustment →

	HN	LN
AL	00	0 - 9

Example for +ve result →

```
sub     AH, AH        ; clear AH
mov     AL, '9'       ; AL := 39H
sub     AL, '3'       ; AL := 39H-33H = 06H
aas                   ; AX := 0006H
or      AL, 30H       ; AL := 36H
```

Example for -ve result →

```
sub     AH, AH        ; clear AH
mov     AL, '3'       ; AL := 33H
sub     AL, '9'       ; AL := 33H-39H = FAH
aas                   ; AX := FF04H
; AH = -1, borrow from tens place, carry sign
; AL = (0 – 9), valid single digit unpacked BCD
; result = (AH×10) + (AL) = -6
```

<u>Arithmetic Instructions</u>

DAS →

Decimal Adjust after Subtraction.
Adjusts result of subtraction of two packed BCD values = packed BCD result.
AL = source and destination operand.

DAS is only useful when it follows
SUB that subtracts one 2-digit, packed BCD value from another and
AL ← byte result.

Adjusts contents of AL = correct 2-digit, packed BCD result.
Decimal borrow → CF and AF = 1.

Lower nibble of AL > 9 →
subtract 06H from AL.
If subtraction sets CF or if upper nibble of AL > 9,
it subtracts 60H from AL.

DAS modifies CF, AF, PF, SF and ZF flags.
OF is not defined after DAS.

Arithmetic Instructions

DAS →

Example →

i) AL = 75
BH = 46
SUB AL, BH ; AL ← 2F = (AL)-(BH), AF = 1
DAS ; AL ← 29 (as F > 9, 2F-06 = 29)

ii) AL = 38
CH = 61
SUB AL, CH ; AL ← D7, CF = 1 (borrow)
DAS ; AL ← 77 (as D>9, D7-60 = 77), CF = 1 (borrow)