ECE 3120
Computer Systems
Arithmetic Programming

Manjeera Jeedigunta
http://blogs.cae.tntech.edu/msjeedigun21
Email: msjeedigun21@tntech.edu
Tel: 931-372-6181, Prescott Hall 120

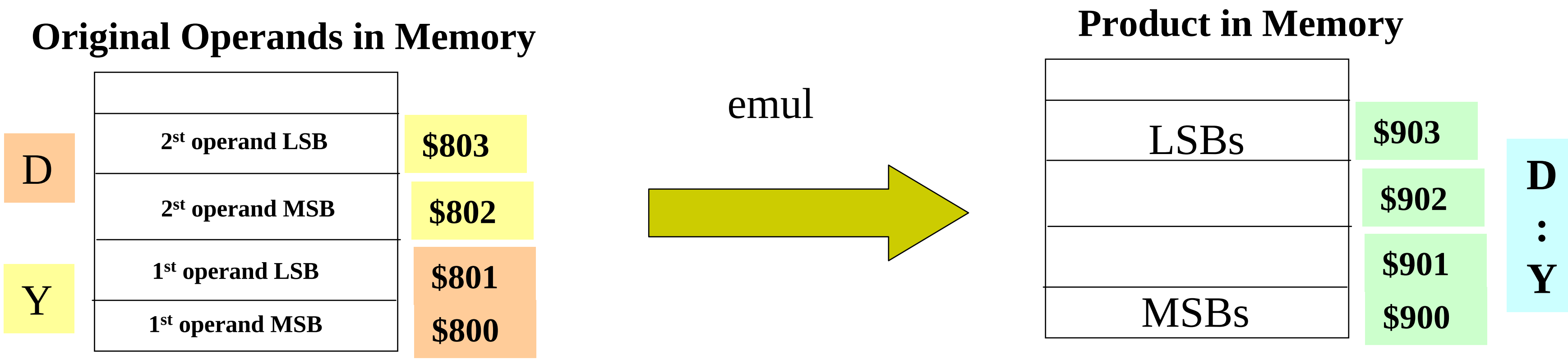
- Today:
  - Multiplication and Division Examples
  - BCD

Multiplication and Division

Table 2.1 Summary of 68HC12 multiply and divide instructions

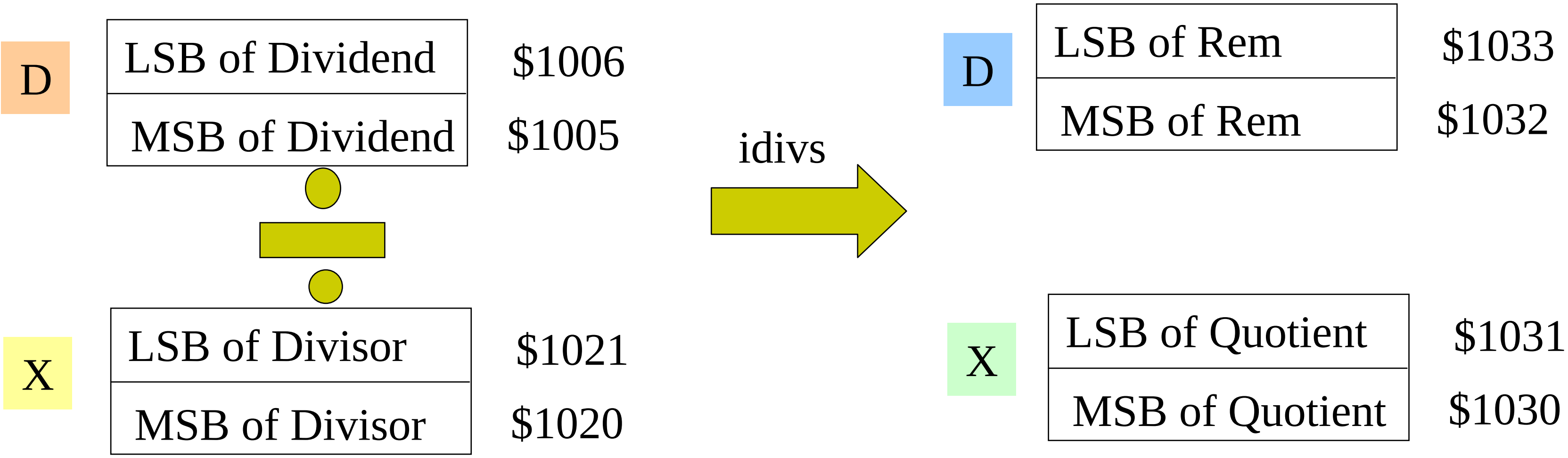
Mnemonic	Function	Operation
EMUL	unsigned 16 by 16 multiply	$(D) \times (Y) \rightarrow$ Y:D
EMULS	signed 16 by 16 multiply	$(D) \times (Y) \rightarrow$ Y:D
MUL	unsigned 8 by 8 multiply	$(A) \times (B) \rightarrow$ A:B
EDIV	unsigned 32 by 16 divide	$(Y{:}D) \div (X)$ quotient $\rightarrow$ Y remainder $\rightarrow$ D
EDIVS	signed 32 by 16 divide	$(Y{:}D) \div (X)$ quotient $\rightarrow$ Y remainder $\rightarrow$ D
FDIV	16 by 16 fractional divide	$(D) \div (X) \rightarrow$ X remainder $\rightarrow$ D
IDIV	unsigned 16 by 16 integer divide	$(D) \div (X) \rightarrow$ X remainder $\rightarrow$ D
IDIVS	signed 16 by 16 integer divide	$(D) \div (X) \rightarrow$ X remainder $\rightarrow$ D

Example 2.10' Write an instruction sequence to multiply the 16-bit numbers stored at $800-$801 and $802-$803 and store the product at $900-$903.




```
ldy     $800        ;load 1st operand into Y
ldd     $802        ;load 2nd operand into D
emul                ; Multiplying the numbers assuming unsigned numbers
sty     $900        ;storing the upper 16 bits
std     $902        ;storing the lower 16 bits
```

Example 2.11 Write an instruction sequence to divide the signed 16-bit number stored at $1005-$1006 by the signed 16-bit number stored at $1020-$1021 and store the quotient and remainder at $1030-$1031 and $1032-$1033, respectively.



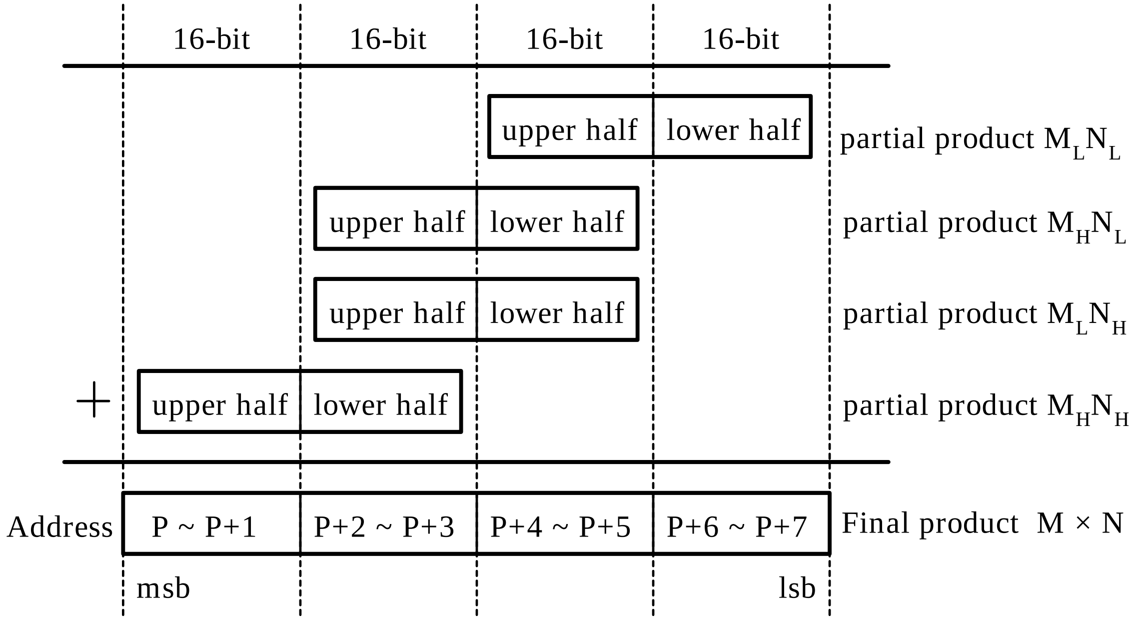

```
ldd     $1005       ;load the dividend into D
ldx     $1020       ;load the divisor into X
idivs               ; perform signed division
stx     $1030       ;storing the quotient
std     $902        ;storing the remainder
```

Illustration of 32-bit by 32-bit Multiplication

- Two 32-bit numbers M and N are divided into two 16-bit halves

$M = M_H M_L$

$N = N_H N_L$



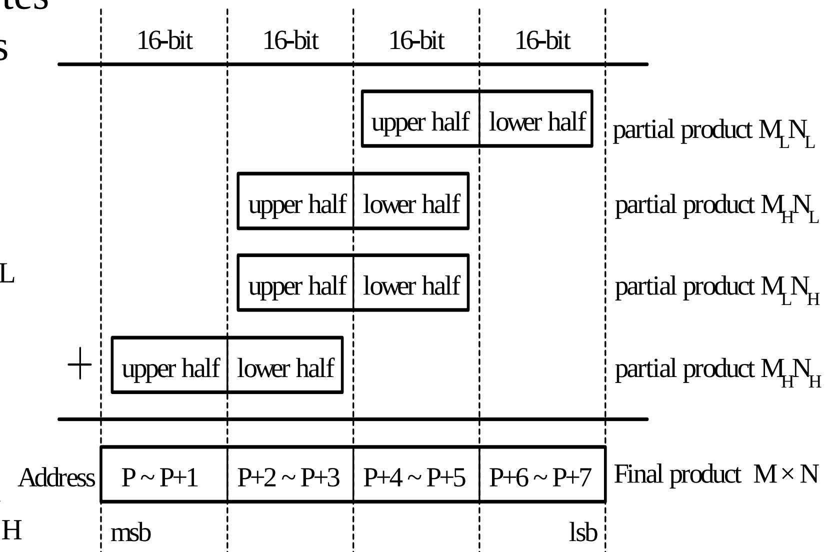
Note: msb stands for most significant byte and lsb for least significant byte

Figure 2.3 Unsigned 32-bit by 32-bit multiplication

Example 2.12 Write a program to multiply two unsigned 32-bit numbers stored at M~M+3 and N~N+3, respectively and store the product at P~P+7.

Solution:

```
        org     $1000
M       ds.b    4               ;Multiplicand 4 bytes
N       ds.b    4               ;Multiplier 4 bytes
P       ds.b    8               ;Product 8 bytes
        org     $1500
        ldd     M+2             ;place ML in D
        ldy     N+2             ;place NL in Y
        emul                    ; compute MLNL
        sty     P+4
        std     P+6
        ldd     M
        ldy     N
        emul                    ; compute MHNH
        sty     P
        std     P+2
        ldd     M
        ldy     N+2
        emul                    ; compute MHNL
```

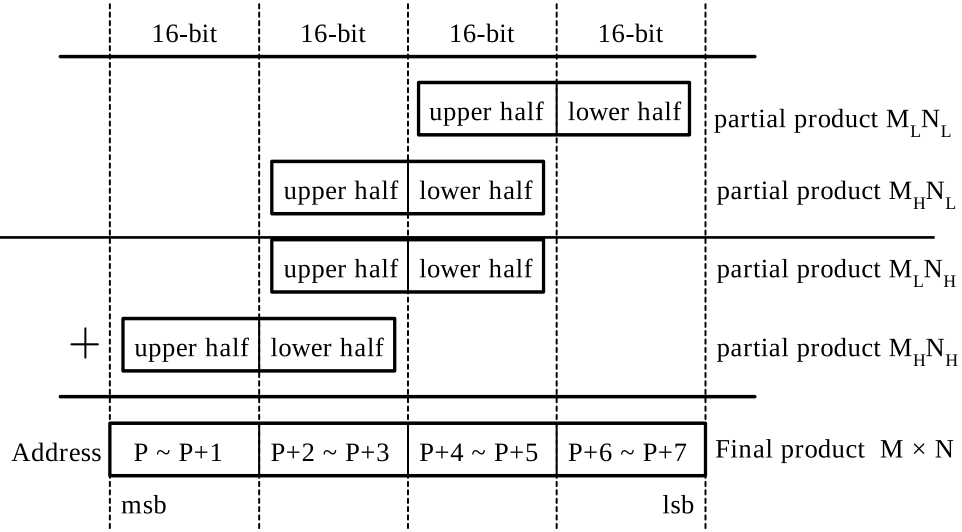


Note: msb stands for most significant byte and lsb for least significant byte

Figure 2.3 Unsigned 32-bit by 32-bit multiplication

Example 2.12 contd…

```
; add MHNL to memory locations P+2~P+5
        addd    P+4
        std     P+4
        tfr     Y,D
        adcb    P+3
        stab    P+3
        adca    P+2
        staa    P+2
; propagate carry to the most significant byte
        ldaa    P+1
        adca    #0          ; add carry to the location at P+1
        staa    P+1         ;         “
        ldaa    P           ; add carry to the location at P
        adca    #0          ;         “
        staa    P           ;         “
; compute MLNH
        ldd     M+2
        ldy     N
        emul
```



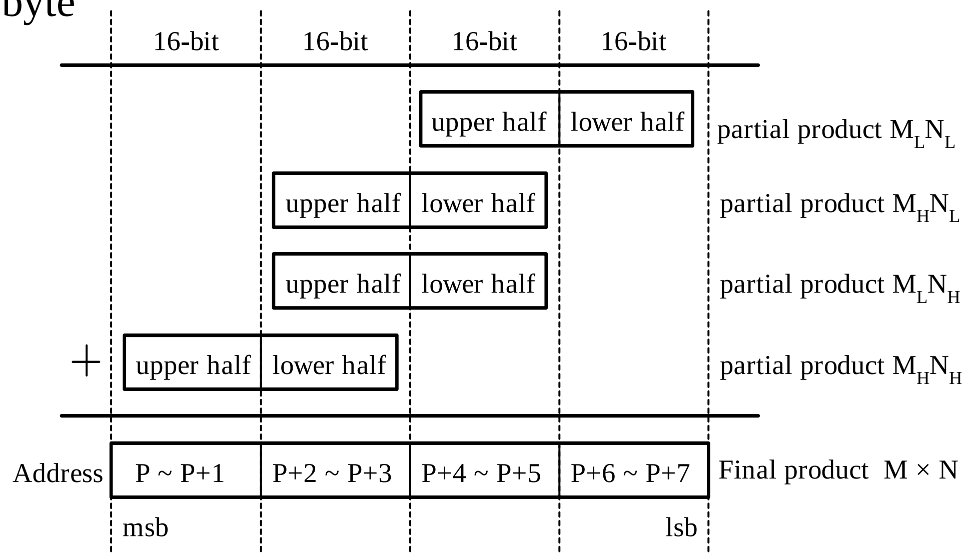
Note: msb stands for most significant byte and lsb for least significant byte

Figure 2.3 Unsigned 32-bit by 32-bit multiplication

Example 2.12 contd..

; add $M_L N_H$ to memory locations P+2 ~ P+5

```
        addd    P+4
        std     P+4
        tfr     Y,D
        adcb    P+3
        stab    P+3
        adca    P+2
        staa    P+2
; propagate carry to the most significant byte
        clra
        adca    P+1
        staa    P+1
        ldaa    P
        adca    #0
        staa    P
        end
```



Note: msb stands for most significant byte and lsb for least significant byte

Figure 2.3 Unsigned 32-bit by 32-bit multiplication

BCD Numbers and Addition

- Each digit is encoded by 4 bits
- Two digits are packed into one byte
- The addition of two BCD numbers is performed by binary addition and an adjust operation using the DAA instruction.
- The instruction DAA can be applied after the instructions ADDA, ADCA, and ABA.
- Simplifies I/O conversion

For example, the instruction sequence

```
LDAA    $800
ADDA    $801
DAA
STAA    $802
```

adds the BCD numbers stored at $800 and $801 and saves the sum at $802.

Example 2.13' Write a program to convert the 16-bit number stored at $800-$801 to BCD format and store the result at $900-$904. Convert each BCD digit into its ASCII code and store it in one byte.

Solution:

- A binary number can be converted to BCD format by using repeated division by 10.
- The largest 16-bit binary number is 65535 which has five decimal digits.
- The first division by 10 obtains the least significant digit, the second division by 10 obtains the second least significant digit, and so on.

```
        org     $800
data    dc.w    12345       ; data to be tested
        org     $900
result  ds.b    5           ; reserve bytes to store the result

        org     $1000
        ldd     data
        ldy     #result
        ldx     #10
        idiv
        addb    #$30        ; convert the digit into ASCII code
        stab    4,Y         ; save the least significant digit
        xgdx
        ldx     #10
```

```
idiv
adcb    #$30
stab    3,Y         ; save the second to least significant digit
xgdx
ldx     #10
idiv
addb    #$30
stab    2,Y         ; save the middle digit
xgdx
ldx     #10
idiv
addb    #$30
stab    1,Y         ; save the second most significant digit
xgdx
addb    #$30
stab    0,Y         ; save the most significant digit
end
```

Next…

- Program Loops
- Read Chapter 2.6