บทที่ 13

การคำนวนทางคณิตศาสตร์ II

ARITHMETIC II Processing ASCII
and BCD Data

วัตถุประสงค์

หลังจากที่ท่านศึกษาบทนี้ท่านจะมีความเข้าใจดังต่อไปนี้

- การคำนวณทางคณิตศาสตร์ของตัวเลข ASCII และ BCD
- รูปแบบของข้อมูลชนิด PACKED และ UNPACKED
- การเปลี่ยนแปลงรูปแบบของตัวเลขชนิดต่างๆ

CT 215 349

บทที่ 13
การคำนวณทางคณิตศาสตร์ II
ARITHMETIC II Processing ASC II and BCD Data

บทนำ

คอมพิวเตอร์มีรูปแบบการคำนวนที่มีประสิทธิภาพมากในรูปแบบการคำนวณเลขฐานสอง ตามตัวอย่างบทที่แล้ว รูปแบบการคำนวณนี้ไม่ใช่ปัญหาหลักในการเขียนโปรแกรมที่ยาวในการกำหนดข้อมูล การป้อนข้อมูลให้กับโปรแกรมจากคีย์บอร์ดในรูปของ ASCII แทนรูปแบบเลขฐานสิบ การแสดงผลที่จอภาพอยู่ในรูปของ ASCII ตัวอย่าง เลขฐานสิบคือ 23 เมื่ออยู่ในรูปเลขฐานสองคือ 00010111 หรือ 17H ในรหัส ASCII ตัวอักขระแต่ละตัวใช้ 1 ไบท์และ ASCII 25 คือค่า 3235H

บทนี้จะอธิบายถึงเทคนิคการเปลี่ยน ASCII เป็นไบนารี สำหรับการคำนวณทางคณิตศาสตร์ และการเปลี่ยนไบนารีเป็น ASCII เพื่อแสดงผล ถ้าท่านเขียนโปรแกรมในภาษาระดับสูง เช่นภาษา C , Pascal ท่านสามารถใช้ Compiler นับตำแหน่งเลขฐานสิบ

รูปแบบ ASCII

ข้อมูลที่ป้อนจากคีย์บอร์ดในรูปของ ASCII ที่เก็บลงในหน่วยความจำของตัวอักษรที่ป้อนเข้ามาค่าของ SAM คือ 53414DH และถ้าแทนค่าตัวเลข 1234 คือ 31323334H แต่วัตถุประสงค์ส่วนมากตัวอักขระจะใช้เป็นข้อมูลที่ป้อนเข้ามาเช่น ชื่อคน หรือ รายการที่ไม่เปลี่ยนแปลง แต่รูปแบบที่มีการคำนวณใช้ตัวเลขเช่น 31323334H รูปแบบต่อไปนี้ใช้ในการคำนวณทางคณิตศาสตร์โดยตรงข้อมูลในรหัส ASCII

AAA ASCII ADJUST FOR ADDITION

AAD ASCII ADJUST FOR DIVISION

AAM ASCII ADJUST FOR MULTIPLICATION

AAS ASCII ADJUST FOR SUBTRACTION

คำสั่งเหล่านี้จะไม่มีโอเปอร์รานด์ และทำการปรับค่าในรีจีสเตอร์ AX โดยอัตโนมัติ การปรับค่าจะเกิดขึ้นเมื่อค่า ASCII อยู่ในรูปของ Unpacked base 10 ขณะที่ตัวโปรเซสเซอร์อยู่ในรูปฐานสอง

การบวก ASCII

พิจารณาผลการบวก ASCII ต่อไปนี้ คือ 8 + 4 หรือ 38H + 34H

```
  3 8 H
+ 3 4 H
  6 C H
```

ผลการบวกจะได้ 6CH ไม่ใช่รหัสที่ถูกต้องของ ASCII ที่เป็นตัวเลข หรือค่าของไบนารี อย่างไรก็ตามเราจะนำค่า 6 มาบวกกับค่าเลขฐานสิบหกทางขวามือ คือ 0CH + 6 = 12H คือคำตอบที่ถูกต้องในเทอมของเลขฐานสิบ ทำไมถึงต้องบวก 6 เพราะว่าค่าแตกต่างระหว่างเลขฐานสิบกับเลขฐานสิบหกห่างกันอยู่ 6 ฉนั้นในการปรับค่าจะใช้คำสั่ง AAA ปรับข้อมูลให้ถูกต้อง

ตัวอย่างสมมุติว่าค่าในรีจีสเตอร์ AX มีข้อมูล 0038H และค่า BX มีค่า 0034H นั่นคือ 38H และ 34H แทนค่าเลข ASCII 2 ตัว ที่นำมาบวกกัน และสามารถใช้คำสั่งปรับค่าดังต่อไปนี้

```
ADD     AL,BL     ;ADD 34h TO 38h
AAA               ;ADJUST FOR ASCII ADD
```

การทำงานของ AAA จะตรวจสอบตัวเลขที่อยู่ขวามือสุด (4 บิต) ของรีจีสเตอร์ AL ถ้าเป็นตัวเลขอยู่ระหว่าง A ถึง F หรือตัวทดช่วยมีค่าเป็น 1 การทำงานก็จะบวก 6 เข้ากับรีจีสเตอร์ AL และบวก 1 เข้ากับรีจีสเตอร์ AH และเซ็ทแฟลกตัวทดและตัวทดช่วยเป็น 1 ในทุกๆกรณีคำสั่ง AAA จะเคลียตัวเลขซ้ายมือของรีจีสเตอร์ AL ผลลัพธ์อยู่ใน AX

```
After the ADD:    006CH
After the AAA:    0102H
```

ถ้าเก็บผลลัพธ์ในรหัส ASCII เราต้องเพิ่ม 3s ในตัวเลขซ้ายสุด โดยใช้คำสั่งต่อไปนี้

```
OR    AX,3030H      ;RESULT NOW 3132
```

การทำงานทั้งหมดนี้เป็นการบวกเลข 1 ไบท์ การบวกหลายๆไบท์ของตัวเลข ASCII ใช้วิธีการวนรอบจากซ้ายไปขวา เริ่มต้นจากบิตที่มีค่าต่ำสุด และเริ่มนับตัวทด ตัวอย่างในรูป 13-1 การบวกขนาด 3 ไบท์ 2 ชุดในรูปของรหัส ASCII คือ ASC1 และ ASC2 ผลบวกขนาด 4 ไบท์เก็บไว้ใน ASCSUM ดังต่อไปนี้ สังเกตการทำงาน

- คำสั่ง CLC เป็นเซ็ตสภาวะของแฟลกตัวทด
- งานในส่วนของ A20 ใช้คำสั่ง ADC สำหรับการบวกเพราะทุกๆการบวกจะต้องรวมตัวทดและบวกให้กับตัวเลขที่อยู่ซ้ายมือ

```
        TITLE ASCADD (com) ADDING ASCII NUMBERS
              .MODEL SMALL
              .CODE
              ORG   100H
BEGIN:        JMP       SHORT MAIN
;-------------------------------------------------------------
ASC1          DB        '578'              ;DATA ITEMS
ASC2          DB        '694'
ASCSUM        DB        '0000'
;-------------------------------------------------------------
MAIN          PROC      NEAR
              CLC                          ;CLEAR CARRY FLAG
              LEA       SI,ASC1+2          ;INITIALIZE ASCII NUMBERS
              LEA       DI,ASC2+2
              LEA       BX,ASCSUM+3
              MOV       CX,3               ;INITIALIZE 3 LOOPS
A20:
              MOV       AH,00              ;CLEAR AH
              MOV       AL,[SI]            ;LOAD ASCII BYTE
              ADC       AL,[DI]            ;ADD WITH CARRY
              AAA                          ;ADLUST FOR ASCII
              MOV       [BX],AL            ;STORE SUM
```

```
        DEC     SI
        DEC     DI
        DEC     BX
        LOOP    A20             ;LOOP 3 TIMES
        MOV     [BX].AH         ;AT END ,STORE CARRY
        LEA     BX,ASCSUM +3    'CONVERT ASCSUM
        MOV     CX,4            ;TO ASCII
A30:
        OR      BYTE PTR[BX],30H
        DEC     BX
        LOOP    A30
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     BEGIN
```

รูป 13-1 ASCII ADDITION

- คำสั่ง MOV จะเคลีย AH ในการทำการวนรอบแต่ละรอบ เพราะคำสั่ง AAA จะบวกค่า 1 กับค่า AH. คำสั่ง ADC จะเป็นการบวกพร้อมตัวทด สังเกตจากการใช้ XOR หรือ SUB ที่ใช้เคลีย AH จะทำให้แฟลกตัวทดเปลี่ยนแปลง
- เมื่อมีการวนรอบเรียบร้อยแล้ว ข้อมูลที่อยู่ใน AH จะเป็น 00 หรือ 01 ในไบท์ซ้ายสุดของ ASCSUM
- ที่จุดสุดท้าย ASCSUM จะมีข้อมูล 01020702 แล้วให้ใส่ ASCII 3s ในแต่ละไบท์โดยใช้โปรแกรมวนรอบผ่าน ASCSUM ในหน่วยความจำ ใช้คำสั่ง OR บวกค่า 30H

การทำงานจะไม่ใช้คำสั่ง OR หลังจากคำสั่ง AAA เพื่อใส่ค่า 3s ทางซ้ายสุด เพราะคำสั่ง OR จะเซ็ตค่าแฟลกตัวทด และเปลี่ยนผลลัพธ์ของการทำงานคำสั่ง ADC ปัญหานี้ก็จะต้องมีการเก็บค่าแฟลกโดยใช้คำสั่ง PUSHF เพื่อเก็บค่าแฟลกรีจีสเตอร์ เพื่อการเอ็กซีคิวส์คำสั่ง OR แล้วดึงข้อมูลกลับมายังแฟลกรีจีสเตอร์อย่างเดิมดังนี้

```
ADC     AL,[DI]      ;ADD WITH CARRY
AAA                  ;ADJUST FOR ASCII
PUSHF                ;SAVE FLAGS
OR      AL,30H       ;INSERT ASCII 3
POPF                 ;RESTORE FLAGS
MOV     [BX],AL      ;STORE SUM
```

คำสั่ง LAHF (Load AH with Flags) และคำสั่ง SAHF (Store AH in Flags) สามารถแทนด้วยคำสั่ง PUSHF และ POPF คำสั่ง LAHF เป็นการโหลดข้อมูล AH กับค่า SF , ZF , AF , PF , CF ส่วนคำสั่ง SAHF เก็บค่าของ AH กลับลงในแฟลก

การลบ ASCII

คำสั่ง AAS (ASCII Adjust for Subtraction) ทำงานเหมือนกับคำสั่ง AAA ส่วนคำสั่ง AAS จะทำการตรวจสอบตัวเลขฐานสิบหกขวามือสุด(4บิต)ของรีจีสเตอร์ AL ถ้าตัวเลขอยู่ในช่วง A ถึง F หรือตัวทดช่วยมีค่าเป็น 1 จะมีการลบ 6 จาก AL ลบ 1 ออกจาก AH และเซ็ตค่า AF และ CF เป็น 1 ส่วนกรณีอื่นๆ จะเคลียตัวเลขซ้ายสุดของ AL เป็น 0

ตัวอย่างต่อไปนี้ 2 ตัวอย่างสมมุติค่า ASC1 = 38H , ASC2 = 34H ตัวอย่างแรกคำสั่ง AAS จะไม่มีการปรับเพราะตัวเลขขวาสุดมีค่าไม่เกิน A

```
                          AX      AF
MOV     AL,ASC1           ;0038
SUB     AL,ASC2           ;0004   0
AAS                       ;0004   0
```

ตัวอย่างที่ 13.1 ตัวเลขขวามือสุดคือ 0CH คำสั่ง AAS จะทำการลบ 6 จาก AL และลบ 1 จาก AH และเซ็ตค่า AF = CF = 1 คำตอบจะมีค่า = -4 หรือเลขฐานสิบหก 0FF06H มันคือค่า 10s คอมพลีเม้นต์

```
                          AX      AF
MOV     AL,ASC2           ;0034
SUB     AL,ASC1           ;00FC   1
AAS                       ;FF06   1
```

การคูณ ASCII

คำสั่ง AAM (ASCII Adjust for Multiplication) เป็นคำสั่งที่ได้จากการปรับผลลัพธ์การคูณของข้อมูล ASCII ในรีจีสเตอร์ AX อย่างไรก็ตามตัวเลขฐานสบหกขวามือสุดจะต้องเคลีย 3s ทิ้งตามข้อมูลให้อยู่ในรูปของ Unpack decimal ตัวอย่าง ASCII มีตัวแรก 31323334 จะกลายเป็น 01020304 ที่จัดอยู่ในรูปของ Unpack Decimal เพราะว่าการปรับจะปรับครั้งละ 1 ไบท์ใน 1 เวลา ท่านสามารถคูณครั้งละ 1 ไบท์ ตามรหัสการทำงานของ Loop

คำสั่ง AAM จะทำการหาร AL ด้วย 10 (0AH) และเก็บผลหารใน AH และเศษไว้ใน AL สำหรับตัวอย่าง AL = 35h และ CL = 39H ถ้ามีการคูณ AL ด้วย CL และเปลี่ยนเป็นรหัส ASCII

```
Instruction             Comment                          AX
AND     CL,0FH          ;CONVERT CL TO 09
AND     AL,0FH          ;COMVERT AL TO 05                0005
MUL     CL              ;MULTIPLY AL BY CL               002D
AAM                     ;CONVERT TO UNPACK DECIMAL       0405
OR      AX,3030H        ;CONVERT TO ASCII                3435
```

การทำงานของคำสั่ง MUL จะได้ค่า 45 (002DH) ใน AX คำสั่ง AAM จะหารด้วย 10 และผลหารอยู่ใน AH = 04 และเศษใน AL = 05 และใช้คำสั่ง OR ให้เปลี่ยนเป็นเลข ASCII ในรูปของ Unpack Decimal

```
        TITLE   ASCMUL  (com) Multiplying ASCII numbers
                .MODEL  SMALL
                .CODE
                ORG     100H
BEGIN:          JMP     MAIN
;-----------------------------------------------------------------
MULTCND         DB      '3783'          ;DATA ITEMS
MULTPLR         DB      '5'
PRODUCT         DB      5 DUP (0)
;-----------------------------------------------------------------
```

```
MAIN    PROC    NEAR
        MOV     CX,04           ;INITIALIZE 4 LOOPS
        LEA     SI,MULTCND+3
        LEA     DI,PRODUCT+4
        AND     MULTPLR,0FH     ;CLEAR ASCII 3
A20:
        MOV     AL,[SI]         ;LOAD ASCII CHAR (OR LODSB)
        AND     AL,0FH          ;CLEAR ASCII 3
        MUL     MULTPLR         ;MULTIPLY
        AAM                     ;ADJUST FOR ASCII
        ADD     AL,[DI]         ;ADD TO
        AAA                     ;STORE
        MOV     [DI],AL         ;PRODUCT
        DEC     DI
        MOV     [DI],AH         ;STORE PRODUCT CARRY
        DEC     SI
        LOOP    A20             ;LOOP 4 TIMES
        LEA     BX,PRODUCT+4    ;CONVERT PRODUCT
        MOV     CX,05           ;TO ASCII
A30:    OR      BYTE PTR[BX],30H
        DEC     BX
        LOOP    A30             ;LOOP 4 TIMES
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     BEGIN
```

รูป 13-2 ASCII Multiplication

การหาร ASCII

คำสั่ง AAD (ASCII Adjust for Division) จะทำการปรับข้อมูลหลังจากการหาร อย่างไรก็ตาม ก่อนการหารจะต้องเคลีย 4 บิตบนของตัวเลข ASCII เพื่อให้อยู่ในรูปแบบ Unpack Decimal คำสั่ง AAD จะต้องมีตัวหารขนาด 2 ไบท์อยู่ใน AX สมมุติว่า AX = 3238H และตัวหารคือ 37H ใน CL มีรูปแบบการปรับดังต่อไปนี้

```
Instruction             Comment                     AX
AND       CL,0FH        ;CONVERT TO UNPACKED
AND       AX,0F0FH      ;CONVERT TO UNPACKED        0208
AAD                     ;CONVERT TO BINARY          001C
DIV       CL            ;DIVIDE BY ?                0004
```

คำสั่ง AAD จะคูณ AH ด้วย 10 และบวกผลคูณ 20 กับค่าใน AL และเคลีย AH = 0 ผลลัพธ์ 001C และเลขฐานสิบจะได้ 28 ตัวหารจะมีขนาดเพียง 1 ไบท์คือ 01 ถึง 09

รูป 13-3 แสดงการหารโดยตัวหารมีขนาด 1 ไบท์ ตัวตั้งมีขนาด 4 ไบท์ ขั้นตอนในการทำงานของตัวตั้งจะทำจากซ้ายไปขวา เศษจะเก็บไว้ใน AH ดังนั้นคำสั่ง AAD จะปรับค่าใน AL ในส่วนสุดท้าย ผลหารจะอยู่ในรูปของ Unpack decimal คือ 00090204 และเศษใน AH = 02

```
        TITLE     ASCDIV  (com) Dividing   ASCII numbers
                  .MODEL  SMALL
                  .CODE
                  ORG     100H
BEGIN:            JMP       SHORT MAIN
;-----------------------------------------------------------------
DIVDND            DB        '3698'          ;DATA ITEMS
DIVISOR           DB        '4'
QUOTNT            DB        4 DUP (0)
;-----------------------------------------------------------------
```

```
MAIN    PROC    NEAR
        MOV     CX,04           ;INITIALIZE 4 LOOPS
        SUB     AH,AH           ;CLEAR LEFT BYTE OF DIVIDEND
        AND     DIVISOR,0FH     ;CLEAR DIVISOR OF ASCII 3
        LEA     SI,DIVDND
        LEA     DI,QUOTNT
A20:
        MOV     AL,[SI]             ;LOAD ASCII CHAR (OR LODSB)
        AND     AL,0FH              ;CLEAR ASCII 3
        AAD                         ;ADJUST FOR DIVIDE
        DIV     DIVISOR             ;DIVIDE
        MOV     [DI],AL             ;STORE QUOTIENT
        INC     SI
        INC     DI
        LOOP    A20                 ;LOOP 4 TIMES
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     BEGIN
```

รูป 13-3 ASCII Division

รูปแบบ BINARY CODE DECIMAL

สำหรับตัวอย่างที่ได้จากการหารผลหารมีค่า 00090204 ถ้าท่านต้องการรวมค่าตัวเลขที่อยู่ทางขวาของแต่ละไบท์ ค่านั้นจะเหลือเพียง 0924 ค่าเหล่านี้จะเก็บอยู่ในรูปของ Binary Code Decimal (BCD) จัดอยู่ในรูปของ PACKED BCD ข้อมูลที่เป็นเลขฐานสิบจะมีตัวเลขเพียง 0-9 ความยาวของ BCD จะมีค่าเพียงครึ่งหนึ่งของตัวเลข ASCII ฉนั้นค่า 0924 คือเลขฐานสิบ ถ้าเปลี่ยนเป็นเลขฐานสิบหกจะได้ 039CH

รูปแบบการบวกการลบของ BCD มีรูปแบบต่อไปนี้

DAA	DECIMAL ADJUSTMENT FOR ADDITION
DAS	DECIMAL ADJUSTMENT FOR SUBTRACTION

การประมวลผลจะทำการครั้งละ 1 ไบท์ ตัวอย่างรูป 13-4 แสดงการบวก BCD โปรแกรมในส่วนของ B10CONV จะทำการเปลี่ยนค่า ASCII ของ ASC1 ,ASC2 เป็น BCD1 ,BCD2 การประมวลผลจำทำจากขวาไปซ้าย ง่ายกว่าที่จะทำซ้ายไปขวา การประมวลผลทำได้ง่ายเพราะเลี่ยน ASCII 1 ไบท์ เป็นตัวเลข BCD ขนาด 1 ไบท์ ส่วนโปรแกรม C10ADD ใช้ Loop ในการบวกเลข BCD ค่าของ BCDSUM จะได้ผลลัพธ์ครั้งสุดท้ายคือ 00127263

```
        TITLE   BCDADD   (COM) CONVERT ASCII TO BCD AND ADD
                .MODEL   SMALL
                .CODE
                ORG      100H
BEGIN:          JMP      SHORT   MAIN
;-------------------------------------------------------------
ASC1            DB       '057836'
ASC2            DB       '069427'
BCD1            DB       '000'
BCD2            DB       '000'
BCDSUM          DB       4 DUP(0)
;-------------------------------------------------------
MAIN            PROC     NEAR
                LEA      SI,ASC1+4      ;INITIALIZE FOR ASC1
                LEA      DI,BCD1+2
                CALL     B10CONV
                LEA      SI,ASC2+4
                LEA      DI,BCD2+2
                CALL     B10CONV
```

```
                CALL    C10ADD
                MOV     AH,4CH
                INT     21H
        MAIN    ENDP
;-------------------------------------------------------
;       CONVERT ASCII TO BCD
;-------------------------------------------------------
    B10CONV     PROC
                MOV     CL,04           ;SHIFT FACTOR
                MOV     DX,03           ;NO. OF WORDS TO CONVERT
    B20:        MOV     AX,[SI]         ;GET ASCII PAIR (LODSW)
                XCHG    AH,AL
                SHL     AL,CL           ;SHIFT OFF
                SHL     AX,CL           ;ASCII 3s
                MOV     [DI],AH         ;STORE BCD DIGITS
                DEC     SI
                DEC     SI
                DEC     DI
                DEC     DX
                JNZ     B20
                RET
        B10CONV ENDP
    ;-----------------------------------------------------
    ;       ADD BCD NUMBERS:
    ;-----------------------------------------------------
    C10ADD      PROC
                XOR     AH,AH           ;CLEAR AH
                LEA     SI,BCD1+2       ;INITIALIZE
                LEA     DI,BCD2+2       ;BCD
                LEA     BX,BCDSUM+3     ;ADDRESSES
                MOV     CX,3
```

```
                CLC
C20:            MOV     AL,[SI]
                ADC     AL,[DI]
                DAA
                MOV     [BX],AL
                DEC     SI
                DEC     DI
                DEC     BX
                LOOP    C20
                RET
C10ADD          ENDP
                END     BEGIN
```

รูป 13-4 BCD Conversion and Arithmetic

รูปแบบการเปลี่ยน ASCII TO BINARY

การคำนวนทางคณิตศาสตร์ในรูปของ ASCII หรือ BCD เหมาะกับการคำนวนตัวเลขน้อยๆ สำหรับการคำนวนส่วนมาก เราใช้ในการเปลี่ยนค่าต่างๆให้อยู่ในรูปของเลขฐานสอง มันง่ายในการเปลี่ยนข้อมูลจาก ASCII โดยตรง และเร็วกว่าการเปลี่ยน ASCII เป็น BCD

วิธีการเปลี่ยนเลข ASCII ฐานสิบเป็นฐานสองมีวิธีดังนี้

- เริ่มต้นจากตัว ASCII ไบท์ขวามือสุด แล้วเลื่อนจากขวาไปซ้าย
- ทำการเคลียตัวเลขฐานสิบหกทางซ้ายในตัวเลข ASCII แต่ละไบท์
- ทำการคูณตัวเลข ASCII แบบก้าวหน้าด้วย 1 , 10 , 100 (1H , 0AH ,64H) ต่อไปเรื่อยๆ แล้วบวกผลคูณ

ตัวอย่าง 13.2 ต่อไปนี้เป็นการเปลี่ยนตัวเลข ASCII 1234 เป็นฐานสอง

step	decimal	hexadecimal
4x1=	4	4
3x10=	30	1E
2X100=	200	C8
1X1000=	1000	3E8
TOTAL:		04D2

การตรวจสอบค่า 04D2 ผลลัพธ์จะเท่ากับ 1234 ในรูป 13-5 เป็นโปรแกรมในการเปลี่ยนเลข ASCII 1234 เป็นค่าของเลขฐานสอง ความยาวของตัวเลข ASCII 4 ตัวเก็บไว้ใน ASCLEN คำสั่งที่กำหนดแอดเดรสเริ่มแรกคือของตัวเลข ASCII คือ ASCVAL-1 อยู่ในรีจีสเตอร์ SI และความยาวของตัวเลขอยู่ใน BX คำสั่งที่ตำแหน่ง B20 จะเคลื่อนย้ายข้อมูลตัวเลข ASCII ไว้ใน AL

```
MOV     AL,[SI + BX]
```

การทำงานใช้แอดเดรส ASCVAL-1 บวกกับข้อมูลใน BX (4) หรือ ASCVAL + 3 (เซ็ตค่าเริ่มแรกไปอยู่ที่ไบท์ขวามสุด) การทำงานในการวนรอบแต่ละครั้งจะลดค่า BX ลงทีละหนึ่ง และอ้างถึงไบท์ต่อไปทางซ้าย สำหรับการกำหนดแอดเดรสแบบนี้ ท่านสามารถใช้ BX แต่ห้ามใช้ CX เพราะ CX ทำงานร่วมกับคำสั่ง LOOP การวนรอบแต่ละครั้งที่ใช้ MULT10 ด้วย 10 ตัวคูณจะเริ่มที่ 1 , 10 , 100 และ ต่อไปเรื่อยๆ ที่จุดสุดท้าย BINVAL จะเป็นข้อมูลฐานสองที่ถูกต้องคือ D204 (เก็บในลักษณะ reverse byte)

```
        TITLE   ASCBIN   (COM) COMVERT ASCII TO BINARY FORMAT
                .MODEL  SMALL
                .CODE
                ORG  100H
BEGIN:          JMP     MAIN
;-------------------------------------------------------------
ASCVAL          DB      '1234'
BINVAL          DW      0
ASCLEN          DW      4
MULT10          DW      1
;-------------------------------------------------------------
MAIN            PROC    NEAR            ;MAIN PROCEDURE
                MOV     CX,10           ;MULT FACTOR
                LEA     SI,ASCVAL-1     ;ADDRESS OF ASCVAL
                MOV     BX,ASCLEN       ;LENGTH OF ASCVAL
B20:            MOV     AL,[SI+BX]      ;SELECT ASCII CHARRACTER
                AND     AX,000FH        ;REMOVE 3-ZONE
                MUL     MULT10          ;MULTIPLY BY 10 FACTOR
```

```
        ADD     BINVAL,AX       ;ADD TO BINARY
        MOV     AX,MULT10       ;CALCULATE NEXT
        MUL     CX              ;10 FACTOR
        MOV     MULT10,AX
        DEC     BX
        JNZ     B20
        MOV     AH,4CH
        INT     21H
MAIN    ENDP
        END     BEGIN
```

รูป 13-5 Conversion of ascii to binary format

รูปแบบการเปลี่ยน BINARY TO ASCII

การพิมพ์ผลลัพธ์ที่ได้จากการคำนวนทางคณิตศาสตร์ ท่านจะต้องเปลี่ยนให้อยู่ในรูปของ ASCII การทำงานจะตรงกันข้ามกับตัวอย่างที่แล้ว แทนที่การคูณก็จะเป็นการหารฐานสองด้วย 10 จนกระทั่งผลลัพธ์น้อยกว่า 10 เศษที่เหลือจะไม่เกิน 9 หรือ 0 - 9 เพื่อจะได้ค่า ASCII ดังตัวอย่างในการเปลี่ยนค่า 4D2H ให้เป็นเลขฐานสิบ

	QUOTIENT	REMAINDER
A 4D2	7B	4
A 7B	C	3
A C	1	2

ผลหารเป็น 1 น้อยกว่าตัวหารคือ A การทำงานก็จะสมบูรณ์ เศษของผลหารคือผลลัพธ์ของ ASCII จากขวาไปซ้าย คือ 1234 ทั้งหมดจะเก็บในหน่วยความจำดังนี้ 31323334

จากรูป 13-6 เป็นโปรแกรมในการเปลี่ยนเลขฐานสองให้อยู่ในรูปของ ASCII แต่ท่านสามารถแสดงขั้นตอนการทำงานของโปรแกรมโดยเอ็กซีคิวส์ที่ละคำสั่งได้ด้วย DEBUG

```
        TITLE   BINASC   (COM) COMVERT BINARY TO ASCII FORMAT
                .MODEL  SMALL
                .CODE
                ORG  100H
BEGIN:          JMP     MAIN
;-------------------------------------------------------
ASCVAL          DB      4 DUP (' ')
BINVAL          DW      04D2H
;-------------------------------------------------------
MAIN            PROC    NEAR            ;MAIN PROCEDURE

                MOV     CX,0010         ;DIVIDE FACTOR
                LEA     SI,ASCVAL+3     ;ADDRESS OF ASCVAL
                MOV     AX,BINVAL       ;GET BINARY FIELD
C20:            CMP     AX,0010         ;VALUE < 10
                JB      C30             ; YES EXIT
                XOR     DX,DX           ;CLEAR UUPER QUOTIENT
                DIV     CX              ;DIVIDE BY 10
                OR      DL,30H
                MOV     [SI],DL         ;STORE ASCII CHARACTER
                DEC     SI
                JMP     C20
C30:            OR      AL,30H          ;STORE LAST QUOTIENT
                MOV     [SI],AL         ; AS ASCII CHARACTER
                MOV     AH,4CH
                INT     21H
MAIN            ENDP
                END     BEGIN
```

รูป 13-6 Conversion of binary to ascii format

ตัวอย่าง 13.3 การเปลี่ยนรหัสตัวเลข ASCII เป็นรหัส PACKED BCD ซึ่งมีอัลกอริทึมต่อไปนี้

```
READ FIRST CHAR
     CASE '_'          :SET SUGN BIT OF BCD BUFFER
          '0' ....'9'  :CONVERT TO BINARY AND PUSH ON STACK
     WHILE     CHAR   <>CR
          READ CHAR
          CASE '0'....'9' :CONVERT TO BINARY AND PUSH ON STACK
     END_WHILE
     REPEAT:
          POP  STACK
          ASSEMBLE 2 DIGITS TO ONE BYTE
     UNTIL ALL DUGITS ARE POPPED
```

โปรแกรมภาษาแอสแซมบลี

```
          READ_INTEGER   PROC
;READ MULTIPLE PRECISION INTEGER NUMBER AND STORE AS
;REAL NUMBER
;INPUT : BX=ADDRESS OF 10 BYTE BUFFER OF 0'S
          XOR  BP,BP     ;BP COUNTS OF NO. DIGITS READ
          MOV  SI,BX     ;COPY OF POUNTER
;READ NUMBER AND PUSH DIGITS ON STACK
          MOV  AH,1      ;READ DIGIT
          INT  21H
;CHECK FOR NEGATIVE
          CMP  AL,'-'
          JNE  RI_LOOP1  ;NOT NEGATIVE
;NEGATIVE, SET SIGN BYTE TO 80H
          MOV  BYTE PTR (BX+9),80H
          INT  21H       ;REDA NEXT CHAR
```

```
;CHECK FOR CR
RI_LOOP1:
          CMP  AL,0DH     ;CR?
          JE   RI_1       ;CR GOTO RI_1
;DIGIT, CONVERT TO BINARY AND SAVE ON STACK
          AND  AL,0FH     ;CONVERT ASCII TO BINARY VALUE
          INC  BP         ;INCREMENT COUNT
          PUSH AX         ;PUSH ON STACK
          MOV  AH,1       ;READ NEXT CHAR
          INT  21H
          JMP  RI_LOOP1   ;REPEAT
;POP NUMBER FROM STACK AND STORE AS PACKED BCD
RI_1:
          MOV  CL,4       ;COUNTER FOR LEFT SHIFT
RI_LOOP2:
          POP  AX         ;LOE DIGIT
          MOV  (BX),AL    ;STORE
          DEC  BP         ;MORE DIGIT
          JE   RI_4       ;NO EXIT LOOP
          POP  AX         ;YES POP HIGH DIGIT
          SHL  AL,CL      ;SHIFT TO HIGH NIBBLE
          OR   (BP),AL    ;STORE
          INC  BX         ;NEXT BYTE
          DEC  BP         ;YES REPEAT
          JG   RI_LOOP2
;CONVERT TO REAL
RI_4:     FBLD TBYTE PTR[SI]  ;LOAD BCD TO 8087 STACK
          FSTB TBYTE PTR[SI]  ;STORE REAL TO MEMORY
          RET
READ_INTEGER ENDP
```

ตัวอย่าง 13.4 การพิมพ์เลข PACKED BCD มีอัลกอริทึมดังต่อไปนี้

```
IF SIGN BIT IS SET , THEN PRINT '_'
GET HIGH ORDER BYTE
     FOR 9 TIMES DO
     CONVERT HIFH BCD DIGIT TO ASCII AND OUTPUT
     CONVERT LOW BCD DIGIT TO ASCII AND OUTPUT
     GET NEXT BYTE
END
```

โปรแกรมภาษาแอสแซมบลี

```
     PRINT_BCD      PROC
;PRINY BCD NUMBET IN BUFFER
;INPUT : BX = ADDRESSES OF 10 BYTE
          TEST BYTE PTR[BX+9],80H       ;CHECK SIGN BIT
          JE   PB_1                     ;POSITIVE, SKIP
          MOV  DL,'-'                   ;NEGATIVE ,OUPUT'-'
          MOV  AH,2
          INT  21H
PB_1:     ADD  BX,8                ;START WITH MSD
          MOV  CH,9                ;9 BYTE
          MOV  CL,4                ;SHIFT 4 TIMES
PB_LOOP:
          MOV  DL,[BX]             ;GET BYTE
          SHR  DL,CL               ;HIGH DIGIT TO LOW NIBBLE
          OR   DL,30H              ;CONVERT TO ASCII
          MOV  AH,2                ;OUTPUT
          INT  21H
          MOV  DL,[BX]             ;GET BYTE AGAIN
          AND  DL,0FH              ;MASK OUT HIGH NIBBLE
          OR   DL,30H              ;CONVERT LOW DIGIT TO ASCII
```

```
        MOV   AH,2
        INT   21H
        DEC   BX                    ;NEXT BYTE
        JG    PB_LOOP
        RET
PRINT_BCD ENDP
```

ตัวอย่าง 13.5 CONVERTING HOURS AND RATE FOR CALCULATING WAGE

โปรแกรมรูป 13-7 เป็นโปรแกรมการคำนวนในการป้อนข้อมูลชั่วโมงการทำงานและอัตราค่าจ้างต่อชั่วโมงของพนักงาน และแสดงผลที่ได้จากการคำนวน มีขั้นตอนดังต่อไปนี้

B10INPT	รับข้อมูลชั่วโมงและอัตราต่อชั่วโมงจากคีย์บอร์ด ค่าที่ป้อนเข้ามาอาจจะเป็นจุดทศนิยมก็ได้
D10HOUR	เป็นการเปลี่ยนค่าตัวเลข ASCII เป็นไบนารี
E10RATE	เป็นการเปลี่ยนค่าตัวเลข ASCII เป็นไบนารี
F10MULT	รูปแบบการคูณ , rounding , shifting
G10WAGE	ใส่ค่าจุดทศนิยมในตำแหน่งทางขวาสุด เริ่มเก็บรหัส ASCII และเปลี่ยนไบนารีเป็น ASCII
K10DISP	เคลียค่าให้เป็นศูนย์และแสดงผลค่าแรง
L10PAUS	ให้ผู้ใช้ดูผลที่ได้จากการคำนวน จนกระทั่งมีการกดคีย์ใดคีย์หนึ่ง การกดคีย์ ESC บอกโปรแกรมหยุดการประมวลผล
M10ASB1	เปลี่ยน ASCII เป็นไบนารี และจำนวนเลขฐานสิบที่ป้อนเข้ามา
Q10SCR	Scrolls the whole screen
Q15WIN	Scrolls window in the middle of the screen where hours,rate and wage

โปรแกรมภาษาแอสแซมบลี

```
        page 60,132
TITLE   SCREMP (EXE)  Enter hours and rate, display wage
        .MODEL SMALL
        .STACK 64
```

```
; ------------------------------------------------
            .DATA
LEFCOL      EQU         28                  ;Equates for screen
RITCOL      EQU         52
TOPROW      EQU         10
BOTROW      EQU         14
HRSPAR      LABEL  BYTE                     ;Hours parameter list:
MAXHLEN      DB         6                   ;----- --------- ----
ACTHLEN      DB         ?
HRSFLD      DB          6 DUP(?)
RATEPAR      LABEL  BYTE                    ;Rate parameter list:
MAXRLEN      DB         6                   ;---- --------- ----
ACTRLEN      DB         ?
RATEFLD      DB         6 DUP(?)
MESSG1      DB          'Hours worked? ','$'
MESSG2      DB          'Rate of pay?  ','$'
MESSG3      DB          'Wage =  '
ASCWAGE      DB         10 DUP(30H), 13, 10,'$'
MESSG4      DB          'Press any key to continue or Esc to quit','$'
ADJUST      DW          ?
BINVAL      DW          00
BINHRS      DW          00
BINRATE     DW          00
COL         DB          00
DECIND      DB          00
MULT10      DW          01
NODEC       DW          00
ROW         DB          00
SHIFT       DW          ?
TENWD       DW          10
; ------------------------------------------------
```

```
            .CODE
BEGIN       PROC    FAR
            MOV         AX,@data                ;Initialize DS
            MOV         DS,AX                   ;  and ES registers
            MOV         ES,AX
            CALL    Q10SCR                      ;Clear screen
A20LOOP:
            CALL    Q15WIN                      ;Clear window
            CALL    Q20CURS                     ;Set cursor
            CALL    B10INPT                     ;Accept hours & rate
            CALL    D10HOUR                     ;Convert hours to binary
            CALL    E10RATE                     ;Convert rate to binary
            CALL    F10MULT                     ;Calculate wage, round
            CALL    G10WAGE                     ;Convert wage to ASCII
            CALL    K10DISP                     ;Display wage
            CALL    L10PAUS                     ;Pause for user
            CMP         AL,1BH                  ;Esc pressed?
            JNE         A20LOOP                 ;  no -- continue
;                                               ;  yes -- end of input
            CALL    Q10SCR                      ;Clear screen
            MOV         AH,4CH
            INT         21H                      ;Exit program
BEGIN       ENDP
;                   Input hours & rate:
;                   ------------------
B10INPT      PROC   NEAR
            MOV         ROW,TOPROW+1            ;Set cursor
            MOV         COL,LEFCOL+3
            CALL    Q20CURS
            INC         ROW
            MOV         AH,09
            LEA         DX,MESSG1               ;Prompt for hours
```

```
                INT         21H
                MOV         AH,0AH
                LEA         DX,HRSPAR           ;Accept hours
                INT         21H
                MOV         COL,LEFCOL+3        ;Set column
                CALL    Q20CURS
                INC         ROW
                MOV         AH,09
                LEA         DX,MESSG2           ;Prompt for rate
                INT         21H
                MOV         AH,0AH
                LEA         DX,RATEPAR          ;Accept rate
                INT         21H
                RET
B10INPT         ENDP
;                           Process hours:
;                           -------------
D10HOUR         PROC    NEAR
                MOV         NODEC,00
                MOV         CL,ACTHLEN
                SUB         CH,CH
                LEA         SI,HRSFLD-1         ;Set right position
                ADD         SI,CX               ;  of hours
                CALL    M10ASBI                 ;Convert to binary
                MOV         AX,BINVAL
                MOV         BINHRS,AX
                RET
D10HOUR         ENDP
;                           Process rate:
;                           ------------
```

```
E10RATE     PROC   NEAR
            MOV        CL,ACTRLEN
            SUB        CH,CH
            LEA        SI,RATEFLD-1          ;Set right position
            ADD        SI,CX                 ;  of rate
            CALL   M10ASBI                   ;Convert to binary
            MOV        AX,BINVAL
            MOV        BINRATE,AX
            RET
E10RATE     ENDP
;                   Multiply, round, and shift:
;                   ---------------------------
F10MULT     PROC   NEAR
            MOV        CX,05
            LEA        DI,ASCWAGE              ;Set ASCII wage
            MOV        AX,3030H                ;  to 30s
            CLD
            REP STOSW
            MOV        SHIFT,10
            MOV        ADJUST,00
            MOV        CX,NODEC
            CMP        CL,06                   ;If more than 6
            JA        F40                      ;  decimals, error
            DEC        CX
            DEC        CX
            JLE        F30                      ;Bypass if 0, 1, 2 decs
            MOV        NODEC,02
            MOV        AX,01
F20:
            MUL        TENWD                    ;Calculate shift factor
            LOOP   F20
            MOV        SHIFT,AX
```

```
                SHR         AX,1                    ;Calculate round value
                MOV         ADJUST,AX
F30:
                MOV         AX,BINHRS
                MUL         BINRATE                 ;Calculate wage
                ADD         AX,ADJUST               ;Round wage
                ADC         DX,00
                CMP         DX,SHIFT                ;Product too large
                JB          F50                     ;  for DIV?
F40:
                SUB         AX,AX
                JMP         F70
F50:
                CMP         ADJUST,00               ;Shift required?
                JZ          F80                     ;  no -- bypass
                DIV         SHIFT                   ;Shift wage
F70:        SUB         DX,DX                       ;Clear remainder
F80:        RET
F10MULT       ENDP
;                           Convert to ASCII:
;                           ----------------
G10WAGE       PROC    NEAR
                LEA         SI,ASCWAGE+7            ;Set decimal point
                MOV         BYTE PTR[SI],'.'
                ADD         SI,NODEC                ;Set right start pos'n
G30:
                CMP         BYTE PTR[SI],'.'
                JNE         G40                     ;Bypass if at dec pos'n
                DEC         SI
G40:
                CMP         DX,00                   ;If DX:AX < 10,
                JNZ         G50
```

```
            CMP         AX,0010                 ;  operation finished
            JB          G60
G50:
            DIV         TENWD                   ;Remainder is ASCII digit
            OR          DL,30H
            MOV         [SI],DL                 ;Store ASCII character
            DEC         SI
            SUB         DX,DX                   ;Clear remainder
            JMP         G30
G60:
            OR          AL,30H                  ;Store last ASCII
            MOV         [SI],AL                 ;  character
            RET
G10WAGE     ENDP
;                       Display wage:
K10DISP     PROC    NEAR
            MOV         COL,LEFCOL+3            ;Set column
            CALL    Q20CURS
            MOV         CX,09
            LEA         SI,ASCWAGE
K20:                                            ;Clear leading zeros
            CMP         BYTE PTR[SI],30H
            JNE         K30                 ;  to blanks
            MOV         BYTE PTR[SI],20H
            INC         SI
            LOOP    K20
K30:
            LEA         DX,MESSG3               ;Display wage
            MOV         AH,09
            INT         21H
            RET
K10DISP     ENDP
```

```
;                       Pause for user:
;                       ---------------
L10PAUS     PROC    NEAR
            MOV         COL,20                  ;Set cursor
            MOV         ROW,22
            CALL    Q20CURS
            MOV         AH,09
            LEA         DX,MESSG4               ;Display pause
            INT         21H
            MOV         AH,00                   ;Accept reply
            INT         16H
            RET
L10PAUS     ENDP
;                       Convert ASCII to binary:
;                       ------------------------
M10ASBI     PROC    NEAR
            MOV         MULT10,0001
            MOV         BINVAL,00
            MOV         DECIND,00
            SUB         BX,BX
M20:
            MOV         AL,[SI]                 ;Get ASCII character
            CMP         AL,'.'                  ;Bypass if dec point
            JNE         M40
            MOV         DECIND,01
            JMP         M90
M40:
            AND         AX,000FH
            MUL         MULT10                  ;Multiply by factor
            ADD         BINVAL,AX               ;Add to binary
            MOV         AX,MULT10               ;Calculate next
            MUL         TENWD                   ;  factor x 10
```

```
                MOV         MULT10,AX
                CMP         DECIND,00           ;Reached decimal point?
                JNZ         M90
                INC         BX                   ;  yes - add to count
M90:
                DEC         SI
                LOOP    M20
                CMP         DECIND,00           ;End of loop
                JZ          M100                ;Any decimal point?
                ADD         NODEC,BX            ;  yes - add to total
M100:           RET
M10ASBI         ENDP
;                       Scroll whole screen:
;                       --------------------
Q10SCR          PROC    NEAR
                MOV         AX,0600H
                MOV         BH,30H              ;Color (07 for BW)
                SUB         CX,CX
                MOV         DX,184FH
                INT         10H
                RET
Q10SCR          ENDP
;                       Scroll display window:
Q15WIN          PROC    NEAR
                MOV         AX,0605H            ;Five rows
                MOV         BH,16H              ;Color (07 for BW)
                MOV         CH,TOPROW
                MOV         CL,LEFCOL
                MOV         DH,BOTROW
                MOV         DL,RITCOL
                INT         10H
                RET
```

```
Q15WIN      ENDP
;                       Set cursor:
;                       ----------
Q20CURS      PROC   NEAR
            MOV         AH,02
            SUB         BH,BH
            MOV         DH,ROW              ;Set row
            MOV         DL,COL              ;Set column
            INT         10H
            RET
Q20CURS      ENDP
            END         BEGIN
```

บทสรุป

- การกำหนดรหัส ASCII ขนาด 1 ไบท์สำหรับฟิลด์ที่เป็นตัวเลขครึ่งไบท์ขวามสุดจะเป็นข้อมูลที่เป็นค่าตัวเลข ครึ่งไบท์ซ้ายมือจะเป็นค่าเลข 3
- การเคลียค่า 3 ของครึ่งไบท์ซ้ายมือเป็น 0 เปลี่ยนฟิลด์ให้อยู่ในรูปของ Unpacked Decimal
- การเก็บตัวเลขรหัส ASCII อยู่ในขนาด 1 ไบท์ ให้อยู่ในรูปของ Packed BCD
- หลังจากการบวกเลขรหัส ASCII ให้ใช้คำสั่ง AAA ในการปรับคำตอบ และหลังจากการลบรหัส ASCII ให้ใช้คำสั่ง AAS ในการปรับคำตอบ
- ก่อนการคูณตัวเลขรหัส ASCII ให้เปลี่ยนตัวตั้งตัวคูณอยู่ในรูปของ Unpacked BCD โดยการเคลียค่า 3 ซ้ายมือเป็น 0 หลังจากการคูณให้ใช้คำสั่ง AAM ในการปรับคำตอบ
- ก่อนการหารตัวเลขรหัส ASCII ให้เปลี่ยนตัวตั้งตัวหารอยู่ในรูปของ Unpacked BCDโดยการเคลียค่า 3 ซ้ายมือเป็น 0 หลังจากการหารให้ใช้คำสั่ง AAD ในการปรับคำตอบ

แบบฝึกหัด

13-1. สมมุติว่าค่า AX = ASCII 9 (0039H) และค่า BX = ASCII 7 (0037H) จงอธิบายผลที่ได้จากคำสั่งต่อไปนี้

(a)
```
ADD   AX,33H
AAA
```

(b)
```
ADD   AX,BX
AAA
```

(c)
```
SUB   AX,BX
AAS
```

(d)
```
SUB   AX,0DH
AAS
```

13-2. ค่าของฟิลด์ตัวเลขชนิด UNPACKED ชื่อ UNPAK มีค่า 01040705 จงเขียนรหัสคำสั่งให้เปลี่ยนค่าเหล่านี้เป็น 31343735H

13-3. ฟิลด์ชื่อ ASCA มีข้อมูลตัวเลข ASCII ค่า 173 และฟิลด์อื่นๆชื่อ ASCB มีข้อมูล ASCII 5 จงเขียนคำสั่งในการคูณตัวเลข ASCII ทั้งสองผลคูณเก็บไว้ใน ASCQUO

13-4. ใช้ชื่อฟิลด์เหมือนกับข้อ 13-3 จงหาร ASCA ด้วย ASCB ผลหารเก็บไว้ที่ ASCQUO

13-5. จงเขียนโปรแกรมในการคูณไบนารีขนาด 16 บิตกับค่า 8 บิต ผลลัพธ์เก็บไว้ในหน่วยความจำ

13-6. จงเขียนโปรแกรมในการหารเลข ASCII โดยตัวตั้งขนาด 2 หลัก ตัวหารขนาด 1 หลัก ผลลัพธ์เก็บในหน่วยความจำ