รายงาน

เรื่อง ทอนเงิน โดยใช้โปรแกรมภาษา Assembly

จัดทำโดย

นายชานนท์ เนตรทอง	รหัสนิสิต 5830250152 S06
นายนเรศ งามเสถียร	รหัสนิสิต 5830250268 S06
นายภัคพล ครุอำโพธิ์	รหัสนิสิต 5830250357 S06

เสนอ

อาจารย์ สุกัญญา ยิ้มงาม

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 01418331 Assembly Language and Computer Architecture

ภาคเรียนที่ 1 ปีการศึกษา 2560

มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

คำนำ

รายงานเรื่อง ทอนเงิน ภาษา Assembly รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา 01418331 Assembly Language and Computer Architecture นิสิตคณะวิทยาศาสตร์ สาขาวิชา วิทยาการคอมพิวเตอร์ มหาวิทยาลัย เกษตรศาสตร์ วิทยาเขตศรีราชา ได้จัดทำขึ้นมาเพื่ออธิบายผลการทำงานของชิ้นงาน โดยเนื้อหาประกอบไปด้วยคำสั่งต่างๆของภาษา Assembly

ผู้จัดทำหวังว่ารายงานฉบับนี้จะเป็นประโยชน์สำหรับผู้ที่สนใจศึกษาเรื่อง การทอนเงิน ในภาษา Assembly ถ้าผิดพลาดประการใด ผู้จัดทำขออภัย ณ ที่นี้ด้วย

คณะผู้จัดทำ

ขั้นตอนในการทำงาน

```
.MODEL SMALL
.DATA

LF   EQU 0AH
CF   EQU 0DH
N DB ?
SUM1 DW 0
SUM2 DW 0

NUM DW 1000
RES  DW 10 DUP ('$')

DIV01 DW 0
SCRAP DW 0
SUMDIV DW 0
```

การเริ่มขนาดของในการจองพื้นที่

- .data ประกาศตัวแปร
- ประกาศตัวแปรโดยให้ LF เป็นตัวแปรตัวที่ 1 ให้เท่ากับขึ้นบรรทัดใหม่
- ประกาศตัวแปรโดยให้ CF เป็นตัวแปรตัวที่ 2 โดยให้ไปขึ้น Cursor แรก
- ประกาศตัวแปรโดยให้ N เป็นตัวว่าง ไว้ใช้ในการเก็บค่าที่รับมา
- ประกาศตัวแปรโดยให้ sum1 เป็นค่าสินค้าที่ต้องจ่าย
- ประกาศตัวแปรโดยให้ sum2 เป็นค่าเงินที่รับจากลูกค้า
- ประกาศตัวแปรโดยให้ num เป็นค่าเก็บผลลัพธ์ของ sum2-sum1
- ประกาศตัวแปรโดยให้ res จองพื้นที่ว่างไว้ 10 ช่อง
- ประกาศตัวแปรโดยให้ DIV01 ไว้ใช้ในการทอนเงิน
- ประกาศตัวแปรโดยให้ SCRAP ไว้เก็บค่าเศษ
- ประกาศตัวแปรโดยให้ SUMDIV เก็บค่า MOD ได้เผื่อใช้ในการเก็บจำนวนในการทอนเหรียญทอนแบงค์

```
MSG1 DB "********* Payment System *********"
     DB CF,LF,"All prices paid -> $"
MSG3 DB CF,LF,LF,"Money 1000 baht change : $"
MSG4 DB CF,LF,"Money 500 baht change : $"
MSG5 DB CF,LF,"Money 100 baht change : $"
MSG6 DB CF,LF,"Money 50 baht change : $"
MSG7 DB CF,LF,"Money 20 baht change : $"
MSG8 DB CF,LF,"Money 10 baht change : $"
MSG9 DB CF,LF,"Money 5 baht change : $"
MSG10 DB CF,LF,"Money 2 baht change : $"
MSG11 DB CF,LF,"Money 1 baht change : $"
MSG12 DB CF,LF,"Money received ( Money>Price) : $"
MSG13 DB CF,LF,LF,"Payment : $"
MSG14 DB CF,LF,LF,"Replay y/n : $"
```

- MSG1 เป็นการแสดงประโยค "********* Payment System *********"

  "All prices paid -> $"

- MSG3 เป็นการแสดงประโยค "Money 1000 baht change : $"
- MSG4 เป็นการแสดงประโยค "Money 500 baht change : $"
- MSG5 เป็นการแสดงประโยค "Money 100 baht change : $"
- MSG6 เป็นการแสดงประโยค "Money 50 baht change : $"
- MSG7 เป็นการแสดงประโยค "Money 20 baht change : $"
- MSG8 เป็นการแสดงประโยค "Money 10 baht change : $"
- MSG9 เป็นการแสดงประโยค "Money 5 baht change : $"
- MSG10 เป็นการแสดงประโยค "Money 2 baht change : $"
- MSG11 เป็นการแสดงประโยค "Money 1 baht change : $"
- MSG12 เป็นการแสดงประโยค "Money received (Money>Price) : $"
- MSG13 เป็นการแสดงประโยค "Payment : $"
- MSG14 เป็นการแสดงประโยค "Replay y/n : $"
- โดย CF เป็นการขึ้น Cuesor ใหม่
- โดย LF เป็นการขึ้นบรรทัดใหม่

```
.CODE
MAIN PROC NEAR
     CALL FUNTION01
     CALL FUNTION02
     CALL PRINT
     CALL Division1000
     CALL Division500
     CALL Division100
     CALL Division50
     CALL Division20
     CALL Division10
     CALL Division5
     CALL Division2
     CALL Division1
     CALL REPLAY
     MOV AH,4CH
     INT 21H
MAIN    ENDP
```

- เป็นการประกาศฟังก์ชัน FUNTION01
- เป็นการประกาศฟังก์ชัน CALL FUNTION02
- เป็นการประกาศฟังก์ชัน CALL PRINT
- เป็นการประกาศฟังก์ชัน CALL Division1000
- เป็นการประกาศฟังก์ชัน CALL Division500
- เป็นการประกาศฟังก์ชัน CALL Division100
- เป็นการประกาศฟังก์ชัน CALL Division50
- เป็นการประกาศฟังก์ชัน CALL Division20
- เป็นการประกาศฟังก์ชัน CALL Division10
- เป็นการประกาศฟังก์ชัน CALL Division5
- เป็นการประกาศฟังก์ชัน CALL Division2
- เป็นการประกาศฟังก์ชัน CALL Division1
- เป็นการประกาศฟังก์ชัน CALL REPLAY

- เมื่อทำงานฟังก์ชั่นทั้งหมดแล้วจะทำการออกโปรแกรม MOV AH,4CH เป็นการเรียก DOS กลับมา
- INT 21H เป็นการเรียกคำสั่ง

```
FUNTION01 PROC NEAR
      MOV AX,@DATA
      MOV DS,AX
      MOV AH,09H
      LEA DX,MSG1
      INT 21H
      MOV DL,0H
FOR:
      MOV       AH,01H
      INT       21H
      CMP       AL,0DH
      JE        END_FOR
      SUB       AL,30H
ฆ     MOV       N,AL
      MOV       AL,10
      MUL       DL
      ADD       AL,N
      MOV       DL,AL
      MOV       BL,DL
      MOV       DH,AH
      MOV       BH,DH
      JMP       FOR
END_FOR:

      MOV       SUM1,BX

      RET
FUNTION01   ENDP
```

- MOV AX,@DATA คือการเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H การแสดงผลข้อความ
- LEA DX,MSG1 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV DL,0H คือ การกำหนดค่าของDLเท่ากับ 0

- FOR: คือชื่อ Lable ที่สร้างไว้ในการวนลูป
- MOV AH,01H ฟังก์ชั่นการรับอักษรตัวเดียว
- INT 21H ดำเนินคำสั่ง
- CMP AL,0DH เปรียบเทียบค่า AL ถ้ากด ENTER
- JE END_FOR ถ้าเท่ากันจะทำการออกจาก FOR
- SUB AL,30H ลบค่าของ AL ออก30H
- MOV N,AL นำค่า AL ไปเก็บไว้ใน N
- MOV AL,10 นำค่า10 ไปเก็บไว้ใน AL
- MUL DL คูณค่า AL ด้วยค่าของ DL
- ADD AL,N บวกค่าALด้วยค่าN
- MOV DL,AL นำค่าAL ไปเก็บไว้ใน DL
- MOV BL,DLนำค่าDL ไปเก็บไว้ในBL
- MOV DH,AH นำค่า AH เก็บไว้ใน DH
- MOV BH,DH นำค่าDH เก็บไว้ในBH
- JMP FOR กระโดดไปยัง FOR
- END_FOR: จบ Lable FOR
- MOV SUM1,BX เก็บค่า BX ไว้ใน SUM1
- RET ส่งค่ากลับ
- FUNTION01 ENDP จบฟังก์ชั่น FUNTION01

```
FUNTION02 PROC NEAR
FRIST:     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG12
     INT 21H
     MOV DL,0H
FOR02:
     MOV       AH,01H
     INT       21H
     CMP       AL,0DH
     JE        END_FOR02
     SUB       AL,30H
     MOV       N,AL
     MOV       AL,10
     MUL       DL
     ADD       AL,N
     MOV       DL,AL
     MOV       BL,DL
     MOV       DH,AH
     MOV       BH,DH
     JMP       FOR02
END_FOR02:

     MOV       SUM2.BX
```

- MOV AX,@DATA คือการเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H การแสดงผลข้อความ
- LEA DX,MSG1คือการเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือการแสดงผลของฟังก์ชั่นข้างบน
- MOV DL,0H คือการกำหนดค่าของDLเท่ากับ 0
- FOR02: คือชื่อ Lable ที่สร้างไว้ในการวนลูป
- MOV AH,01H ฟังก์ชั่นการรับอักษรตัวเดียว
- INT 21H ดำเนินคำสั่ง
- CMP AL,0DH เปรียบเทียบค่า AL ถ้ากด ENTER
- JE END_FOR02 ถ้าเท่ากันจะทำการออกจาก FOR02
- SUB AL,30H ลบค่าของ AL ออก30H

- MOV N,AL นำค่า ALไปเก็บไว้ใน N
- MOV AL,10 นำค่า10 ไปเก็บไว้ใน AL
- MUL DL คูณค่า AL ด้วยค่าของ DL
- ADD AL,N บวกค่าALด้วยค่าN
- MOV DL,AL นำค่าALไปเก็บไว้ใน DL
- MOV BL,DLนำค่าDLไปเก็บไว้ในBL
- MOV DH,AH นำค่า AH เก็บไว้ใน DH
- MOV BH,DH นำค่าDH เก็บไว้ในBH
- JMP FOR02 กระโดดไปยัง FOR02
- END_FOR02: จบ Lable FOR02
- MOV SUM2,BX เก็บค่า BX ไว้ใน SUM1

```
    CMP BX,SUM1
    JB FRIST

    MOV AX,SUM2
    MOV BX,SUM1
    SUB AX,BX
    MOV NUM,AX
    MOV DIV01,AX

    MOV AX,@DATA
    MOV DS,AX
    MOV AH,09H
    LEA DX,MSG13
    INT 21H

    RET
FUNTION02  ENDP
```

- CMP BX,SUM1 : การเทียบค่า BX กับ SUM1
- JB FRIST : ถ้า BX น้อยกว่า SUM1 จะไปยัง Lable FRIST
- MOV AX,SUM2 : เก็บค่า SUM2 ไว้ในAX

- MOV BX,SUM1 : เก็บค่า SUM1 ไว้ในBX
- SUB AX,BX คือ AX = AX - BX
- MOV NUM,AX นำค่า AX เก็บไว้ใน NUM
- MOV DIV01,AX นำค่า AX เก็บไว้ใน DIV01

- MOV AX,@DATA : การเข้าถึง data segment
- MOV DS,AX : เก็บค่า AX ไว้ใน DS
- MOV AH,09H : แสดงผลข้อความ
- LEA DX,MSG13 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- FUNTION02 ENDP : จบฟังก์ชั่น FUNTION02

```
;-----------------------
Division1000 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG3
     INT 21H
     MOV BX,1000
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division1000 ENDP
;
```

- Division1000 PROC NEAR : ชื่อฟังก์ชั่น Division1000
- MOV AX,@DATA คือ การเข้าถึง data segment

- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG3 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV BX,1000 : เก็บค่า 1000 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01ไว้ใน AX
- DIV BX : หาร AX ด้วย BX
- MOV SCRAP,DX : เก็บค่าของ DX ที่เป็นเศษจากการหารมาเก็บใน SCRAP
- ADD AX,48 : บวก AX ด้วย 48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงผลอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division1000 ENDP : จบฟังก์ชั่น Division1000

```
Division500 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG4
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,500
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division500 ENDP
```

- Division500 PROC NEAR ชื่อฟังก์ชั่น Division500
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG4 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ในAX
- MOV DIV01,AX : เก็บค่าAXไว้ใน DIV01
- MOV BX,500 : เก็บค่า500ไว้ในBX
- MOV DX,0 : เก็บค่า0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01ใน AX
- DIV BX : หารค่าAXด้วยBX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP

- ADD AX,48 : บวกค่า AX ด้วย 48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division500 ENDP : จบฟังก์ชั่น Division500

```
Division100 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG5
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,100
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division100 ENDP
```

- Division100 PROC NEAR : ชื่อฟังก์ชั่น Division100
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG5 คือ การเข้าถึงข้อความที่จะแสดงผล

- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ในAX

- MOV DIV01,AX : เก็บค่าAXไว้ใน DIV01
- MOV BX,100 : เก็บค่า100ไว้ในBX
- MOV DX,0 : เก็บค่า0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01ใน AX
- DIV BX : หารค่าAXด้วยBX

- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP
- ADD AX,48 : บวกค่าAXด้วย48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว

- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division100 ENDP : จบฟังก์ชั่น Division100

```
Division50 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG6
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,50
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division50 ENDP
```

- Division50 PROC NEAR : ชื่อฟังก์ชั่น Division50
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG6 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,50 : เก็บค่า 50 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01 ใน AX
- DIV BX : หารค่า AX ด้วย BX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP

- ADD AX,48 : บวกค่าAXด้วย48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division50 ENDP : จบฟังก์ชั่น Division50

```
Division20 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG7
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,20
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division20 ENDP
```

- Division20 PROC NEAR : ชื่อฟังก์ชั่น Division20
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG7 คือ การเข้าถึงข้อความที่จะแสดงผล

- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,20 : เก็บค่า 20 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01 ใน AX
- DIV BX : หารค่า AX ด้วย BX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP
- ADD AX,48 : บวกค่า AX ด้วย 48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division20 ENDP : จบฟังก์ชั่น Division20

```
Division10 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG8
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,10
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division10 ENDP
```

- Division10 PROC NEAR : ชื่อฟังก์ชั่น Division10
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG8 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,10 : เก็บค่า 10 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01 ใน AX
- DIV BX : หารค่า AX ด้วย BX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP
- ADD AX,48 : บวกค่า AX ด้วย 48

- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division10 ENDP : จบฟังก์ชั่น Division10

```
Division5 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG9
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,5
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division5 ENDP
```

- Division10 PROC NEAR : ชื่อฟังก์ชั่น Division10
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG9 คือ การเข้าถึงข้อความที่จะแสดงผล

- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,5 : เก็บค่า 5 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01 ใน AX
- DIV BX : หารค่า AX ด้วย BX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP
- ADD AX,48 : บวกค่า AX ด้วย 48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division5 ENDP จบฟังก์ชั่น Division5

```
Division2 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG10
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,2
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division2 ENDP
```

- Division2 PROC NEAR : ชื่อฟังก์ชั่น Division2
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ
- LEA DX,MSG10 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,2 : เก็บค่า 2 ไว้ใน BX
- MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01 ใน AX
- DIV BX : หารค่าAXด้วยBX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP

- ADD AX,48 : บวกค่า AX ด้วย 48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division2 ENDP : จบฟังก์ชั่น Division2

```
Division1 PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG11
     INT 21H
     MOV AX,SCRAP

     MOV DIV01,AX
     MOV BX,1
     MOV DX,0
     MOV AX,DIV01
     DIV BX

     MOV SCRAP,DX
     ADD AX,48
     MOV SUMDIV,AX

     MOV AH,02
     MOV DX,SUMDIV
     INT 21H
     RET
Division1 ENDP
```

- Division2 PROC NEAR : ชื่อฟังก์ชั่น Division2
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX
- MOV AH,09H : การแสดงผลข้อความ

- LEA DX,MSG11 คือ การเข้าถึงข้อความที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- MOV AX,SCRAP : นำค่า SCRAP เก็บไว้ใน AX
- MOV DIV01,AX : เก็บค่า AX ไว้ใน DIV01
- MOV BX,1 : เก็บค่า1ไว้ในBX
- MOV DX,0 : เก็บค่า0 ไว้ใน DX
- MOV AX,DIV01 : เก็บค่า DIV01ใน AX
- DIV BX : หารค่าAXด้วยBX
- MOV SCRAP,DX : เก็บค่า DX ไว้ใน SCRAP
- ADD AX,48 : บวกค่าAXด้วย48
- MOV SUMDIV,AX : เก็บค่า AX ไว้ใน SUMDIV
- MOV AH,02 : แสดงอักษรตัวเดียว
- MOV DX,SUMDIV : เข้าถึงตัวแปรที่จะแสดงผล
- INT 21H คือ การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- Division1 ENDP : จบฟังก์ชั่น Division1

```
PRINT PROC NEAR
    MOV AX,@DATA
    MOV DS,AX

    MOV AH,0
    MOV AX,NUM

    LEA SI,RES
    CALL HEX2DEC

    LEA DX,RES
    MOV AH,9
    INT 21H

    RET
PRINT ENDP
```

- PRINT PROC NEAR : ชื่อฟังก์ชั่น PRINT
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX : เก็บค่า AX ไว้ใน DS
- MOV AH,0 : เก็บค่า 0 ไว้ใน AH
- MOV AX,NUM : เก็บค่า NUM ไว้ใน AX
- LEA SI,RES : ผูก SI กับ RES
- CALL HEX2DEC : เรียกใช้ฟังก์ชั่นเปลี่ยนเลขฐาน 16 เป็นเลขฐาน 10
- LEA DX,RES : เข้าถึงตัวแปรที่จะแสดงผล
- MOV AH,9 : การแสดงผลข้อความ
- INT 21H : การแสดงผลของฟังก์ชั่นข้างบน
- RET : ส่งค่ากลับ
- PRINT ENDP : จบฟังก์ชั่น PRINT

```
HEX2DEC PROC NEAR
     MOV CX,0
     MOV BX,10

LOOP1:  MOV DX,0
        DIV BX
        ADD DL,30H
        PUSH DX
        INC CX
        CMP AX,9
        JG LOOP1

        ADD AL,30H
        MOV [SI],AL

LOOP2:  POP AX
        INC SI
        MOV [SI],AL
        LOOP LOOP2
        RET

HEX2DEC ENDP
```

- HEX2DEC PROC NEAR : ชื่อฟังก์ชั่น HEX2DEC
- MOV CX,0 : เก็บค่า 0 ไว้ใน CX
- MOV BX,10 : เก็บค่า 10 ไว้ใน BX
- LOOP1: MOV DX,0 : เก็บค่า 0 ไว้ใน DX
- DIV BX : หารด้วย BX
- ADD DL,30H : บวก DL ด้วย 30H
- PUSH DX : ดึง DX
- INC CX : บวกค่า CX อีก 1
- CMP AX,9 : เปรียบเทียบค่า AX กับ 9
- JG LOOP1 : ถ้า AX มากกว่า 9 จะกลับไปที่ LOOP1
- ADD AL,30H : บวกค่า AL ด้วย 30H
- MOV [SI],AL : เก็บค่า AL ไว้ใน [SI]
- LOOP2: POP AX : ใส่AX
- INC SI : เพิ่มค่า SI อีก 1

- MOV [SI],AL : เก็บค่า AL ใน [SI]
- LOOP LOOP2 : ลดค่า CX ถ้า CX เท่ากับ 0 จะออกจาก LOOP2
- RET : ส่งค่ากลับ
- HEX2DEC ENDP : จบฟังก์ชั่น HEX2DEC

```
REPLAY PROC NEAR
     MOV AX,@DATA
     MOV DS,AX
     MOV AH,09H
     LEA DX,MSG14
     INT 21H
     MOV AH,01H
     INT 21H
     MOV AH,0

     CMP AL,'y'
     JE REPLAY01
     JNE ENDRE
REPLAY01 :
          CALL CLRSCR
          CALL MAIN

ENDRE:   RET
REPLAY ENDP
;--------------------
```

- REPLAY PROC NEAR : ชื่อฟังก์ชั่น REPLAY
- MOV AX,@DATA คือ การเข้าถึง data segment
- MOV DS,AX : เก็บค่า AX ไว้ใน DS
- MOV AH,09H : แสดงผลข้อความ
- LEA DX,MSG14 : เรียกข้อความ
- INT 21H : ดำเนินคำสั่ง
- MOV AH,01H : รับเข้าอักษร 1 ตัว
- INT 21H : ดำเนินคำสั่ง
- MOV AH,0 : เก็บค่า 0 ไว้ใน AH

- CMP AL,'y' : เปรียบเทียบค่า AL กับตัวอักษร y
- JE REPLAY01 : ถ้าเท่ากันจะไปยัง REPLAY01
- JNE ENDRE : ถ้าไม่เท่ากันจะไปยัง ENDRE
- REPLAY01
- CALL CLRSCR : เรียกใช้ฟังก์ชั่น CLRSCR
- CALL MAIN : เรียกใช้ฟังก์ชั่น MAIN
- ENDRE: RET : ส่งค่าคืน
- REPLAY ENDP : จบฟังก์ชั่น REPLAY

```
clrscr proc NEAR
     mov ax,0003h
     int 10h
     ret
clrscr endp
;-----------------
END MAIN
```

- clrscr proc NEAR : ชื่อฟังก์ชั่น clrscr
- mov ax,0003h คือ ฟังก์ชั่นการเคลียร์หน้าจอ
- int 10h : ทำการเคลียร์หน้าจอ
- ret : ส่งค่าคืน
- clrscr endp : จบฟังก์ชั่น clrscr
- END MAIN : จบฟังก์ชั่น MAIN

ผล Run ของ Code

```
********* Payment System *********
All prices paid -> 700
Money received ( Money>Price): 2500

Payment: 1800

Money 1000 baht change: 1
Money 500 baht change: 1
Money 100 baht change: 3
Money 50 baht change: 0
Money 20 baht change: 0
Money 10 baht change: 0
Money 5 baht change: 0
Money 2 baht change: 0
Money 1 baht change: 0

Replay y/n:
```

- All prices paid เป็นราคาสินค้าที่ต้องจ่าย = 700 บาท
- Money received เป็นเงินที่ได้รับการจ่าย = 2500 บาท
- Payment เป็นการชำระเงิน = 1800 บาท
- จะเป็นการทอนเงิน 1000 บาท = 1 ใบ
- จะเป็นการทอนเงิน 500 บาท = 1 ใบ
- จะเป็นการทอนเงิน 100 บาท = 3 ใบ
- Replay ถ้าทำต่อก็กด y ไม่ทำรายการต่อก็กด n