ข้อที่ 8

```
 .data
   .balign 4
get_A: .asciz "A : "
   .balign 4
get_B: .asciz "B : "
   .balign 4
pattern: .asciz "%d"
   .balign 4
output: .asciz "%d mod %d = %d\n"

   .balign 4
A:  .word   0
   .balign 4
B:  .word   0
   .balign 4
result: .word 0

@ Variables to backup link register
.balign 4
lr_bu: .word 0
.balign 4
lr_bu_2: .word 0

.text
mod_func:
   @ Save (Store) Link Register to lr_bu_2
   LDR R7, addr_lr_bu_2
   STR lr,[R7] @ Mem[addr_lr_bu_2] <- LR

      MOV R6,#0 @ A>=0 , R6 = 0
      CMP R1,#0 @cmp A,0
      BGE else1 @if A>=0 to else1
      MOV R6,#1 @ A<0 , R6=1
      MOV R5,#-1
      MUL R1,R1,R5 @ A=A*(-1)
   else1:
      CMP R2,#0 @cmp B,0
      BGE end @if B >= 0 to end
      MOV R5,#-1
      MUL R2,R2,R5
   end:
   while:
      CMP R1,R2 @cmp A,B
      BLT endW @if A<B to endW
      SUB R1,R1,R2 @A=A-B
      B while
   endW:
      CMP R6,#0
      BEQ pos @if A = 0 to pos (A is positive)
      MOV R3,R1
      MOV R4,#-1
      MUL R3,R3,R4 @R3=R3*(-1) ,result
      B endRe
   pos:
      MOV R3,R1 @result
   endRe:

   @ Load Link Register from back up 2
   LDR lr, addr_lr_bu_2
   LDR lr, [lr] @ LR <- Mem[addr_lr_bu_2]

   BX lr

@ address of Link Register back up 2
addr_lr_bu_2: .word lr_bu_2

   .global main
main:

   @ Store (back up) Link Register
   LDR R1, addr_lr_bu
   STR lr, [R1] @ Mem[addr_lr_bu] <- LR

   @ Print A :
   LDR R0, addr_get_A
   BL printf

   @ Get A from user via keyboard
   LDR R0, addr_pattern
   LDR R1, addr_A
   BL scanf

   @ Print B :
   LDR R0, addr_get_B
```

```
    BL printf

    @ Get B from user via keyboard
    LDR R0, addr_pattern
    LDR R1, addr_B
    BL scanf

    LDR R1, addr_A
    LDR R1, [R1]
    LDR R2, addr_B
    LDR R2, [R2]
    BL mod_func

    LDR R0,=output
    LDR R1, addr_A
    LDR R1, [R1]
    LDR R2, addr_B
    LDR R2, [R2]
    BL printf

    @ Restore Link Register to return
    LDR lr, addr_lr_bu
    LDR lr, [lr] @ LR <- Mem[addr_lr_bu]
    BX lr


addr_get_A: .word get_A
addr_get_B: .word get_B
addr_pattern: .word pattern
addr_output: .word output
addr_A: .word A
addr_B: .word B
addr_lr_bu: .word lr_bu

.global printf
.global scanf
```

ข้อที่ 9

```
   .data
   .balign 4
get_A: .asciz "A : "
   .balign 4
get_B: .asciz "B : "
   .balign 4
pattern: .asciz "%d"
   .balign 4
output: .asciz "GCD of %d and %d = %d\n"

   .balign 4
A:  .word   0
   .balign 4
B:  .word   0
   .balign 4
result: .word 0

@ Variables to backup link register
.balign 4
lr_bu: .word 0

.balign 4
lr_bu_2: .word 0
```

```
.text
gcd_func:
   @ Save (Store) Link Register to lr_bu_2
   LDR R7, addr_lr_bu_2
   STR lr,[R7] @ Mem[addr_lr_bu_2] <- LR

      MOV R6,#0 @ A>=0 , R6 = 0

      CMP R1,#0 @cmp A,0
      BGE else1 @if A>=0 to else1
      MOV R6,#1 @ A<0 , R6=1
      MOV R5,#-1
      MUL R1,R1,R5 @ A=A*(-1)
   else1:
      CMP R2,#0 @cmp B,0
      BGE end @if B >= 0 to end
      MOV R5,#-1
      MUL R2,R2,R5
   end:

   gcd:
      CMP R1,R2 @cmp A,B
      BEQ endGcd @if A=B to endGcd
      CMP R1,R2 @cmp A,B
      BLE elseGcd @if A<=B to elseGcd
      SUB R1,R1,R2 @A=A-B if A>B
      b gcd
   elseGcd:
      SUB R2,R2,R1 @B=B-A if B>A
      b gcd
   endGcd:
      MOV R3,R1
   @ Load Link Register from back up 2
   LDR lr, addr_lr_bu_2
```

```
    LDR lr, [lr] @ LR <- Mem[addr_lr_bu_2]
    BX lr

@ address of Link Register back up 2
addr_lr_bu_2: .word lr_bu_2


    .global main
main:

    @ Store (back up) Link Register
    LDR R1, addr_lr_bu
    STR lr, [R1] @ Mem[addr_lr_bu] <- LR

    @ Print A :
    LDR R0, addr_get_A
    BL printf

    @ Get A from user via keyboard
    LDR R0, addr_pattern
    LDR R1, addr_A
    BL scanf

    @ Print B :
    LDR R0, addr_get_B
    BL printf

    @ Get B from user via keyboard
    LDR R0, addr_pattern
    LDR R1, addr_B
    BL scanf

    LDR R1, addr_A
    LDR R1, [R1]
    LDR R2, addr_B
    LDR R2, [R2]
    BL gcd_func

    @print output
    LDR R0,=output
    LDR R1, addr_A
    LDR R1, [R1]
    LDR R2, addr_B
    LDR R2, [R2]
    BL printf

    @ Restore Link Register to return
    LDR lr, addr_lr_bu
    LDR lr, [lr] @ LR <- Mem[addr_lr_bu]
    BX lr


addr_get_A: .word get_A
addr_get_B: .word get_B
addr_pattern: .word pattern
addr_output: .word output
addr_A: .word A
addr_B: .word B
addr_lr_bu: .word lr_bu

.global printf
.global scanf
```