```
; File: fibo.o
;---------------------------------------------------------------------
; The present file tests the global possibilities for the ASSEMBLER /
;---------------------------------------------------------------------

        XDEF fiboCount, counter, Fibonacci, main   ;XDEF: This directive is used to declare external references, meaning that these labels or symbols can be accessed from other files.

myData: section         ; This starts a new data section named myData.ds.w 1:
fiboCount:    ds.w  1    ; This directive is used to reserve space for a word (typically 2 bytes). Here, space is reserved for fiboCount and counter.
counter:      ds.w  1

locData: section        ; locData: section: This starts another data section named locData
fib1:         ds.w  1    ; Again, space is reserved for fib1, fib2, fibo, and i.
fib2:         ds.w  1
fibo:         ds.w  1
i:          ds.w  1

myCode: section          ; myCode: section: This begins the section of the program where the actual instructions (code) reside.
Fibonacci:
    PSHD              ; Push D register to the stack
    CLRB              ; Clear B register
    CLRA              ; Clear A register
    STD   fib1        ; Store D register (which is now 0) to fib1
    INCB              ; Increment B register (now B is 1)
    STD   fib2        ; Store D register (which has value 1) to fib2
    LDX   0,SP          ; Load stack pointer into X register
    STX   fibo        ; Store X register value to fibo
    LDAB  #2            ; Load value 2 into AB register
    STD   i           ; Store D register value (2) to i
    BRA   cond          ; Branch always to label "cond". Until here, the code initializes variables and sets up the environment to cpmpute Fib sequence.
  loop:
    LDD   fib1          ; Load fib1 into D register
    ADDD  fib2          ; Add fib2 to D register
    STD   fibo          ; Store the result in fibo
    LDX   fib2          ; Load fib2 into X register
    STX   fib1          ; Store X register value to fib1
    STD   fib2          ; Store D register value to fib2 (update fib2 with the previously calculated Fibonacci number)
    LDX   i           ; Load i into X register
    INX               ; Increment X register
    STX   i           ; Store incremented X register value back to i

  cond:
    LDD   i           ; Load i into D register
    CPD   0, SP         ; Compare D register with value at stack pointer
    BLS   loop          ; Branch to "loop" if result is less than or same

    LDD   fibo          ; Load fibo into D register
    LEAS  2, SP         ; Adjust stack pointer by adding 2
```

```
    RTS             ; Return from subroutine


main:
    LDS   #$3FF0      ; initialize Stack
    CLR   fiboCount    ; Clear fiboCount
   mainLoop:
    CLRB            ; Clear B register
    CLRA            ; Clear A register
    STD   counter     ; Store 0 in counter
    STD   +0,SP       ; Push D register (0) to the stack
    BRA   testEnd     ; Branch always to label "testEnd"
  incCnt:
    LDD   counter     ; Load counter into D register
    ADDD  #1          ; Add 1 to D register
    STD   counter     ; Store updated D register value to counter
    BSR   Fibonacci   ; Call Fibonacci subroutine
    STD   fiboCount   ; Store the Fibonacci number in fiboCount
    LDX   +0,SP       ; Load value from stack into X register
    INX             ; Increment X register
    STX   +0,SP       ; Store incremented X register value back to stack
  testEnd:
    LDD   +0,SP       ; Load value from stack into D register
    CPD   #48         ; Compare D register with 48
    BLE   incCnt      ; Branch to "incCnt" if result is less than or equal
    BRA  mainLoop     ; Otherwise, branch to "mainLoop"
    PULD            ; Pop value from stack into D register
    RTS             ; Return from subroutine
```

Output: Observe **Fibo,**