

```

; 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
    CLR B            ; 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 compute 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,

fibonacciCount	5 int	fibonacciCount	1 int	fibonacciCount	1 int
counter	1 int	counter	2 int	counter	3 int
fib1	0 int	fib1	1 int	fib1	1 int
fib2	1 int	fib2	1 int	fib2	2 int
fibonacci	1 int	fibonacci	1 int	fibonacci	2 int
i	2 int	i	3 int	i	4 int

  

fibonacciCount	3 int	fibonacciCount	2 int	fibonacciCount	5 int
counter	5 int	counter	4 int	counter	6 int
fib1	3 int	fib1	2 int	fib1	5 int
fib2	5 int	fib2	3 int	fib2	8 int
fibonacci	5 int	fibonacci	3 int	fibonacci	8 int
i	6 int	i	5 int	i	7 int

  

fibonacciCount	8 int	fibonacciCount	13 int	fibonacciCount	21 int
counter	7 int	counter	8 int	counter	9 int
fib1	8 int	fib1	13 int	fib1	21 int
fib2	13 int	fib2	21 int	fib2	34 int
fibonacci	13 int	fibonacci	21 int	fibonacci	34 int
i	8 int	i	9 int	i	10 int

  

fibonacciCount	34 int	fibonacciCount	55 int	fibonacciCount	89 int
counter	10 int	counter	11 int	counter	12 int
fib1	34 int	fib1	55 int	fib1	89 int
fib2	55 int	fib2	89 int	fib2	144 int
fibonacci	55 int	fibonacci	89 int	fibonacci	144 int
i	11 int	i	12 int	i	13 int