ANDREW NGUYEN PROJECT 2 README
COMPUTER ARCHITECTURE: 211

Objective:
To create a program displaying the value of the fibonacci sequence in x86 Assembly Language. With the given information f(n) = f(n-1) + f(n-2) n >= 2 and f(0) = 0, f(1) = 1.

Approach:

First wrote the program as a recursive function. When I tested out program, the computer took too long to find the number in the sequence when I choose something like 30.

Design:
Create a fibonacci routine called fib(); I wrote it in c first to calculate the fibonacci series in a for loop. A routine addTwo(); to add 2 integer numbers with an overflow check. If the addition of the two numbers overflow than it will return -1. Otherwise it will return the fibonacci number. To check for overflow logic in the addition of the two numbers in the sequence. The program will report an overflow when the summation of the number turns negative.

Assembly:
In the first part, the program initializes the stack and set up the registers for temporary storage. The program statically assigns the first 2 terms and return that initial values which are 0 and 1. The loop counter are initialized here.

```
fib:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $40, %esp
        movl    $0, -8(%ebp)
        movl    $1, -4(%ebp)
        cmpl    $1, 8(%ebp)
        jg      .L2
        movl    8(%ebp), %eax
        movl    %eax, -20(%ebp)
        jmp     .L4
```

In the loop .L2, set index variable to 0 and jump to loop .L5

```
.L2:
        movl    $0, -12(%ebp)
        jmp     .L5
```

In loop .L5, the variable n is moved to register %eax and subtracts 2. The variable n-2 is compared with the the index variable to see if it greater than or equal to zero. If it is go to loop .L6. Otherwise save the result and terminate the loop.

```
.L5:
        movl   8(%ebp), %eax
        subl   $2, %eax
        cmpl   -12(%ebp), %eax
        jge    .L6
        movl   -16(%ebp), %eax
        movl   %eax, -20(%ebp)
```

In loop .L6, the 2 fibonacci terms gets passed to the addTwo routine. The addTwo should return a value and save it in a register. If the return value is equal to -1 then we have the overflow condition and return the -1 value and exit the loop at .L4. When the return value is not equal to -1(overflow) then continue the calculation in .L7

```
.L6:
        movl   -4(%ebp), %eax
        movl   %eax, 4(%esp)
        movl   -8(%ebp), %eax
        movl   %eax, (%esp)
        call   addTwo
        movl   %eax, -16(%ebp)
        cmpl   $-1, -16(%ebp)
        jne    .L7
        movl   $-1, -20(%ebp)
        jmp    .L4
```

In the loop .L7, add the two fibonacci terms and save continue the loop in .L5.

```
.L7:
        movl   -4(%ebp), %eax
        movl   %eax, -8(%ebp)
        movl   -16(%ebp), %eax
        movl   %eax, -4(%ebp)
        addl   $1, -12(%ebp)
```

In .L4, save the result value the register %eax and return.

```
.L4:
        movl   -20(%ebp), %eax
        leave
        ret
```

In addTwo routine, it add two numbers and save it to %eax. Then it saves it to -4 (%ebp). It compares this value with the first input term in the routine. If sum is less than the first input term then it an overflow condition and return -1. If that is not true then the

program will compare if sum is less than the second term. If sum is less than the second input term then it an overflow condition and return -1.

```
addTwo:
        pushl  %ebp
        movl   %esp, %ebp
        subl   $20, %esp
        movl   12(%ebp), %eax
        addl   8(%ebp), %eax
        movl   %eax, -4(%ebp)
        movl   -4(%ebp), %eax
        cmpl   8(%ebp), %eax
        jge    .L12
        movl   $-1, -20(%ebp)
        jmp    .L14
```

If the sum continues to not be an overflow then it proceeds to .L15 and returns the sum.

```
.L15:
        movl   -4(%ebp), %eax
        movl   %eax, -20(%ebp)
```

C Program:
Check for user input. If the input equals to “-h” the print out the help line.
Else, the program expects the input string has the n term value. It then converts into a signed integer and save it to n.

```
if(argc==2) {

        if (strcmp( argv[1], "-h") == 0){
        printf("Usage: fibonacci [-h] <non-zero integer>\n");
        return 0;
        }
        else {
        n = atoi(argv[1]);
        }
}
```

If data is not provided then program ask the value for the n term. which is a signed integer value. Ask the user for an the f(n) where f(n) is the n term of the fibonacci series. It then calls the fib routine to calculate the fibonacci series. If the return value is -1 then report overflow. Otherwise report the value of the fibonacci sum.

```
printf("Input value for Fibonacci Sequence:");
        scanf("%d", &n);
        }
        n = fib(n);
        if ( n == -1 ) {
                printf("Overflow\n");
                return -1;
        }
         printf("%d\n\n", n);
```

How to use the program:
To execute the program properly you can either type in fibonacci or “fibonacci #” where # equals the sequence number you want to get the value of.

Included Files:
Makefile
fib.h
fib.s
fibonacci.c
Readme.pdf