CS232 Lab 3 Report

Rijul Bhat (22B0971)

1 Program 1

First the assembler contents (in x86) of the given object file were observed, this can be done easily using the shell command

```
objdump -D program1 -M intel
```

In the output we got main function along with many library funcions.

Program1: main

```
0000000000401146 <main>:
  401146:       55                      push   rbp
  401147:       48 89 e5                mov    rbp,rsp
  40114a:       48 83 ec 30             sub    rsp,0x30
  40114e:       89 7d dc                mov    DWORD PTR [rbp-0x24],edi
  401151:       48 89 75 d0             mov    QWORD PTR [rbp-0x30],rsi
  401155:       bf 08 20 40 00          mov    edi,0x402008
  40115a:       b8 00 00 00 00          mov    eax,0x0
  40115f:       e8 dc fe ff ff          call   401040 <printf@plt>
  401164:       c7 45 fc 00 00 00 00    mov    DWORD PTR [rbp-0x4],0x0
  40116b:       c7 45 f4 00 00 00 00    mov    DWORD PTR [rbp-0xc],0x0
  401172:       c6 45 f3 01             mov    BYTE PTR [rbp-0xd],0x1
  401176:       eb 67                   jmp    4011df <main+0x99>
  401178:       83 45 f4 01             add    DWORD PTR [rbp-0xc],0x1
  40117c:       83 7d f4 01             cmp    DWORD PTR [rbp-0xc],0x1
  401180:       7e 45                   jle    4011c7 <main+0x81>
  401182:       8b 45 f8                mov    eax,DWORD PTR [rbp-0x8]
  401185:       89 45 ec                mov    DWORD PTR [rbp-0x14],eax
  401188:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  40118b:       48 98                   cdqe
  40118d:       8b 4c 85 e4             mov    ecx,DWORD PTR [rbp+rax*4-0x1c]
  401191:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  401194:       8d 50 01                lea    edx,[rax+0x1]
  401197:       89 d0                   mov    eax,edx
  401199:       c1 f8 1f                sar    eax,0x1f
  40119c:       c1 e8 1f                shr    eax,0x1f
  40119f:       01 c2                   add    edx,eax
  4011a1:       83 e2 01                and    edx,0x1
  4011a4:       29 c2                   sub    edx,eax
  4011a6:       89 d0                   mov    eax,edx
  4011a8:       48 98                   cdqe
  4011aa:       8b 44 85 e4             mov    eax,DWORD PTR [rbp+rax*4-0x1c]
  4011ae:       29 c1                   sub    ecx,eax
  4011b0:       89 ca                   mov    edx,ecx
  4011b2:       89 55 f8                mov    DWORD PTR [rbp-0x8],edx
  4011b5:       83 7d f4 02             cmp    DWORD PTR [rbp-0xc],0x2
  4011b9:       7e 0c                   jle    4011c7 <main+0x81>
  4011bb:       8b 45 ec                mov    eax,DWORD PTR [rbp-0x14]
  4011be:       3b 45 f8                cmp    eax,DWORD PTR [rbp-0x8]
  4011c1:       74 04                   je     4011c7 <main+0x81>
  4011c3:       c6 45 f3 00             mov    BYTE PTR [rbp-0xd],0x0
  4011c7:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  4011ca:       8d 50 01                lea    edx,[rax+0x1]
  4011cd:       89 d0                   mov    eax,edx
  4011cf:       c1 f8 1f                sar    eax,0x1f
  4011d2:       c1 e8 1f                shr    eax,0x1f
  4011d5:       01 c2                   add    edx,eax
  4011d7:       83 e2 01                and    edx,0x1
  4011da:       29 c2                   sub    edx,eax
  4011dc:       89 55 fc                mov    DWORD PTR [rbp-0x4],edx
  4011df:       8b 45 fc                mov    eax,DWORD PTR [rbp-0x4]
  4011e2:       48 98                   cdqe
  4011e4:       48 8d 14 85 00 00 00    lea    rdx,[rax*4+0x0]
  4011eb:       00
  4011ec:       48 8d 45 e4             lea    rax,[rbp-0x1c]
  4011f0:       48 01 d0                add    rax,rdx
  4011f3:       48 89 c6                mov    rsi,rax
  4011f6:       bf 40 20 40 00          mov    edi,0x402040
  4011fb:       b8 00 00 00 00          mov    eax,0x0
  401200:       e8 4b fe ff ff          call   401050 <__isoc99_scanf@plt>
  401205:       83 f8 01                cmp    eax,0x1
  401208:       0f 84 6a ff ff ff       je     401178 <main+0x32>
  40120e:       83 7d f4 02             cmp    DWORD PTR [rbp-0xc],0x2
  401212:       7f 11                   jg     401225 <main+0xdf>
  401214:       bf 48 20 40 00          mov    edi,0x402048
  401219:       e8 12 fe ff ff          call   401030 <puts@plt>
  40121e:       b8 ff ff ff ff          mov    eax,0xffffffff
  401223:       eb 21                   jmp    401246 <main+0x100>
  401225:       80 7d f3 00             cmp    BYTE PTR [rbp-0xd],0x0
  401229:       74 0c                   je     401237 <main+0xf1>
  40122b:       bf 77 20 40 00          mov    edi,0x402077
  401230:       e8 fb fd ff ff          call   401030 <puts@plt>
  401235:       eb 0a                   jmp    401241 <main+0xfb>
  401237:       bf 7b 20 40 00          mov    edi,0x40207b
  40123c:       e8 ef fd ff ff          call   401030 <puts@plt>
  401241:       b8 00 00 00 00          mov    eax,0x0
  401246:       c9                      leave
  401247:       c3                      ret
  401248:       0f 1f 84 00 00 00 00    nop    DWORD PTR [rax+rax*1+0x0]
  40124f:       00
```

Analysing main, first few lines just push base pointer onto the stack and makes stack pointer the new base pointer. Then command for printing ”Enter three or more numbers (Terminate with CTRL + D):” is given and rbp-0x4 and rbp-0xc are initialised to 0 (4 bytes) and rbp-0xd is initialised to 0x1 (1 byte). Next we encounter a jump statement to line 4011df, where eax is set equal to rbp-0x4 (4 bytes) and the msb of eax is filled in rax using the cdqe command. Then input is read and if the input is valid then jump to 401178 and increment rbp-0xc by 1. So here we can observe a while loop is getting formed where rbp-0xc will hold the number of inputs received until now. Further if rbp-0xc is greater than 1 then continue else jump to 4011c7. So we jumped to 4011c7 as right now rbp-0xc stores 1. Then some operations are performed on eax and edx which is equivalent to

```
edx = eax + 1
eax = ((eax + 1 + (eax + 1) s>> 15 u>> 15)  &  1) - ((eax + 1) s>> 15 u>> 15)
```

where s>> and u>> represent signed and unsigned bit shift respectively.
These operations complement the value in rbp-0x4 and store them again in rbp-0x4. Then new input is taken and stored in rax*4+rbp-0x1c. rax stores either 0 or 1 all the time so it determines where the incoming input goes i.e. either to rpb-0x1c and rbp-0x18. Next there is an if condition where it is checked if rbp-0xc is greater than 1 less than 2 then difference of values goes in rbp-0x8 and loop takes next input storing it in ecx, else if rbp-0xc is 2 jump to 4011c7 or else compare the difference stored in rbp-0x14 and rbp-0x8. if rbp-0x14 == rbp-0x8 jump to 4011c7 (continue

taking input) else rbp-0xd = 0. Once rbp-0xd is set to 0 it is confirmed that difference is not equal and NO will get output, if rbp-0xd does not get changed throughout that is if it stays 1 until inputs are taken implies that the difference remained constant and it will output YES.

2 Program 2

First the assembler contents (in x86) of the given object file were observed, this can be done easily using the shell comand

`objdump -D program2 -M intel`

In the output we got two functions, main and func along with many library functions.

Program2: main and func

```
0000000000401136 <func>:
  401136:       55                      push   rbp
  401137:       48 89 e5                mov    rbp,rsp
  40113a:       53                      push   rbx
  40113b:       48 83 ec 28             sub    rsp,0x28
  40113f:       48 89 7d d8             mov    QWORD PTR [rbp-0x28],rdi
  401143:       48 83 7d d8 00          cmp    QWORD PTR [rbp-0x28],0x0
  401148:       75 07                   jne    401151 <func+0x1b>
  40114a:       b8 01 00 00 00          mov    eax,0x1
  40114f:       eb 50                   jmp    4011a1 <func+0x6b>
  401151:       48 c7 45 e8 00 00 00    mov    QWORD PTR [rbp-0x18],0x0
  401158:       00
  401159:       48 c7 45 e0 01 00 00    mov    QWORD PTR [rbp-0x20],0x1
  401160:       00
  401161:       eb 30                   jmp    401193 <func+0x5d>
  401163:       48 8b 45 e0             mov    rax,QWORD PTR [rbp-0x20]
  401167:       48 83 e8 01             sub    rax,0x1
  40116b:       48 89 c7                mov    rdi,rax
  40116e:       e8 c3 ff ff ff          call   401136 <func>
  401173:       48 89 c3                mov    rbx,rax
  401176:       48 8b 45 d8             mov    rax,QWORD PTR [rbp-0x28]
  40117a:       48 2b 45 e0             sub    rax,QWORD PTR [rbp-0x20]
  40117e:       48 89 c7                mov    rdi,rax
  401181:       e8 b0 ff ff ff          call   401136 <func>
  401186:       48 0f af c3             imul   rax,rbx
  40118a:       48 01 45 e8             add    QWORD PTR [rbp-0x18],rax
  40118e:       48 83 45 e0 01          add    QWORD PTR [rbp-0x20],0x1
  401193:       48 8b 45 e0             mov    rax,QWORD PTR [rbp-0x20]
  401197:       48 39 45 d8             cmp    QWORD PTR [rbp-0x28],rax
  40119b:       73 c6                   jae    401163 <func+0x2d>
  40119d:       48 8b 45 e8             mov    rax,QWORD PTR [rbp-0x18]
  4011a1:       48 8b 5d f8             mov    rbx,QWORD PTR [rbp-0x8]
  4011a5:       c9                      leave
  4011a6:       c3                      ret

00000000004011a7 <main>:
  4011a7:       55                      push   rbp
  4011a8:       48 89 e5                mov    rbp,rsp
  4011ab:       48 83 ec 10             sub    rsp,0x10
  4011af:       bf 08 20 40 00          mov    edi,0x402008
  4011b4:       b8 00 00 00 00          mov    eax,0x0
  4011b9:       e8 72 fe ff ff          call   401030 <printf@plt>
  4011be:       48 8d 45 f8             lea    rax,[rbp-0x8]
  4011c2:       48 89 c6                mov    rsi,rax
  4011c5:       bf 27 20 40 00          mov    edi,0x402027
  4011ca:       b8 00 00 00 00          mov    eax,0x0
  4011cf:       e8 6c fe ff ff          call   401040 <__isoc99_scanf@plt>
  4011d4:       48 8b 45 f8             mov    rax,QWORD PTR [rbp-0x8]
  4011d8:       48 89 c7                mov    rdi,rax
  4011db:       e8 56 ff ff ff          call   401136 <func>
  4011e0:       48 89 c6                mov    rsi,rax
  4011e3:       bf 2c 20 40 00          mov    edi,0x40202c
  4011e8:       b8 00 00 00 00          mov    eax,0x0
  4011ed:       e8 3e fe ff ff          call   401030 <printf@plt>
  4011f2:       b8 00 00 00 00          mov    eax,0x0
  4011f7:       c9                      leave
  4011f8:       c3                      ret
  4011f9:       0f 1f 80 00 00 00 00    nop    DWORD PTR [rax+0x0]
```

Analysing main, first few lines just push base pointer onto the stack and makes stack pointer the new base pointer. Then command for printing ”Enter a non-negative integer:” is given and input is read and stored in rbp-0x8, which is then stored in rax. QWORD PTR is used to only load 64 bits from rbp-0x8 to rax. Then this input is stored in rdi implies it is acting as the argument to the function call in the next line(40011db).

Analysing func, Here again, first few lines are used to push rsp, rbp to the stack. Also rsp is maintained 28 bytes above rbp. Next, the argument of this function is stored in rbp-0x28. Then we can observe the presence of a conditional statement, where rbp-0x28 (current argument of func) is compared with 0. If true then go to 4011a1, which like a continue/return statement as the function terminates after that line. Else store 0 into rbp-0x18 and store 1 into rbp-0x20. Then there is a jump command to 401193 where value of rbp-0x20 is copied into rax. Then a check is being made, instructing PC to jump to 401163 if rbp-0x28 >= rbp-0x20 else the function heads towards termination. We can see that if we jump to 401163 then there is a function call present in which argument is given to be rbp-0x20 - 1 and the return value is stored in rbx. Then there is another function call with argument rbp-0x28 - rbp-0x20. The value returned by this function is multiplied by previous return value and added to rbp-0x18, where it will be stored.

The value returned by the function called by main is then stored in rsi and it is printed. The above analysis can be presented in the form of psuedo-code:

Algorithm 1: Pseudocode Program2

```
func func($n$):
    sum ← 0;
    for $i$ : $1$ to $n$ do
        sum ← $f(i-1)f(n-i)+$ sum;
    end
    return sum;
read integer n;
print f(n);
```

From the pseudocode it is easy to observe what program 2 is trying to achieve. It is a well known sequence of numbers call the Catalan numbers. They are given by the recursive relation $a_n = \sum_{i=0}^{n-1} a_{n-1-i}a_i$ and have solution $a_n = \dfrac{\binom{2n}{n}}{n+1}$.