# Report CS232 Lab3 Part 1

Arohan Hazarika(22B3948)

October 2023

## 1 Program 1 Q1

- I have analysed the main function for this program. We have the rbp base pointer pointing to the base of the current stack frame and rsp is the stack pointer pointing to the top of the stack. We first print "Enter three or more numbers" calling the printf function.
- We define variables at locations rbp-0x4, rbp-0xc and rbp-0xd with values 0, 0 and 1 respectively. There is an unconditional jump to the memory address 4011df where we load eax(originally loaded with 0) with the value of rbp-0x4.
- cdqe instruction extends a DWORD(32-bit value) in the EAX register to a QWORD(64-bit value) in the RAX register. Then rsi is made to store input to be obtained from scanf function call(its value is [rbp-0x1c+rax*4]. I have guessed that eax has value 1 if we give a number as input(that is not "CTRL+D")
- Thereafter there is a conditional statement, if eax has value stored 1 we jump to memory address 401178, we then increment our variable stored in rbp-0xc(counter) by 1, if the value of counter after increment is less than or equal to 1, we jump to memory address 4011c7(*), and we perform **operation** on eax and edx registers. Since value at the address rbp-0x4 is 0 ,we make edx's value to be 1 after the **operation** and update the value at rbp-0x4 location. After this we take the input again and repeat the previous point.
- (*)Else we load the value stored in rbp-0x4 into eax(just before this, its loaded from rbp-0x8 and loaded into rbp-0x14) and then give the instruction cdqe. ecx is loaded with the input. Then we do an **operation** on eax and edx.
- Note: Operation: Case when value of rbp-0x4=0 gives value in eax=value in edx=1; Case when value of rbp-0x4=1 gives value in eax=value in edx=0.

- Thereafter again cdqe and load the value stored in [rbp+rax*4-0x1c](thats the previous input) into eax and store its difference with the current input in ecx and then edx is loaded with this difference. We load the difference into the variable rbp-0x8 and finally compare the value stored in rbp-0xc to 2, if its value is less than or equal to 2, we go and read one more input. We now first jump to 401178 and then using temporary variable we transfer the value stored in temp variable at rbp-0x8 to temp variable at rbp-0x14. Then pass through jump statement since counter is greater than 2.
- We then do the above process of calculating the difference of third and second and storing it in (rbp-0x8) temp variable.Then we compare the values of rbp-0x8 and rbp-0x14 .If they are equal we go and check for further input and keep rbp-0xd value as 1.If they are not equal anytime during the procedure the rbp-0xd value is made as 0. We keep doing this till all inputs are over considering we have more than equal to 3 inputs. Now if we finish all inputs we exit out of loop.
- We exit if after scanf call, eax doesn't have the value 1. If rbp-0xc(counter) has the value greater than 2(#) we jump to memory address 401225 where there is a comparison whether rbp-0xd has value stored 1 or not, if it is 1 "YES" else its "NO".
- (#) If counter is less equal to than 2, then its printed "You have not entered enough numbers, try again".
- This program is checking whether a sequence of numbers entered is an AP sequence or not.

## 2 Program 2 Q1

- For this program, I analysed the main and func function. At start, we define the rbp base pointer which is defined for every new function call.
- We then print the line "Enter a non-negative integer:" by calling the printf function. After that we load the address where the scanf input is stored(its stored in rbp-0x8). We load the scanf input into rax register which is further loaded into rdi register. rdi is used to give the function argument to the function "func". Just after doing so we store 0 in rax regsiter.
- Now I analysed the function "func". Here we again define the rbp and rsp pointers as we do for any new function and thereafter load the input argument into rbp-0x28. Base case is checked by comparing the input argument with zero, if true there is a direct jump to memory address 4011a1(and also 1 is loaded in eax register) and return value 1. Else it is sent to the 401151 memory address.

- After going to 401151 address, we define two variables rbp-0x18 and rbp-0x20 and load values 0 and 1 respectively which actually represent a sum and an iterator(for a loop). There is an unconditional jump to memory address 401193 where we load rax with the value stored in iterator(rbp-0x20). Subtract 1 from rax and then load rdi with the value of rax as rdi will now act as an argument for the next recursive call of func.

- After we return, rax stores the returned value of func just called before which we an see anyways later which is again moved to rbx to store the returned value $f(i-1)$. Then rax is loaded with the input of the current function stored in rbp-0x28. Iterator(i) value is subtracted from rax and then again rax value is moved to rdi for recursive function call.

- Now the returned value $f(n-i)$ is multiplied with $f(i-1)$ which stored by rbx and the result is loaded to rax which is added to rbp-0x18(the summation). After that iterator(rbp-0x20) is incremented by 1.

- Unconditional jump to 401163, where rbp-0x28 value (iterator) is compared with input input argument rbp-0x28, this is the condition of the loop as if this case is true we again jump to memory address 401163 and go through the loop once again. If the condition is false, which means iterator value becomes greater than input argument of the function, then we exit the loop.

- Finally rax is loaded with the summation rbp-0x18 and return call takes place.

- Now in the main function back, load the returned value of rax into rsi. edi is used for printing the output by calling the printf function.

- The recursive equation used in this problem is

$$f(n+1) = \sum_{i=0}^{n} f(i)f(n-i)$$

with $f(0) = 1$