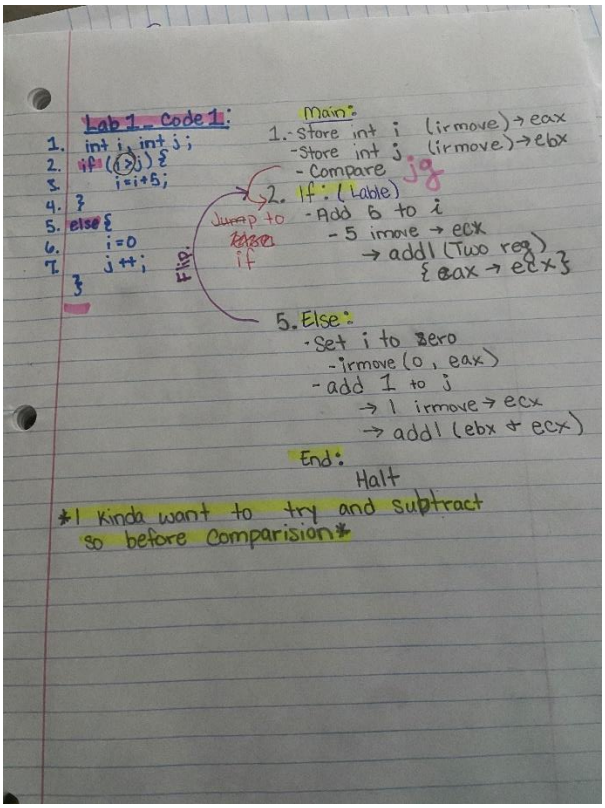
PROBLEM 1 AND 2 NOTE:

I had problems reopening the .yo files once they were created so I copied and pasted them into text files, I also copied and pasted them down below just in case.

Problem 1 :



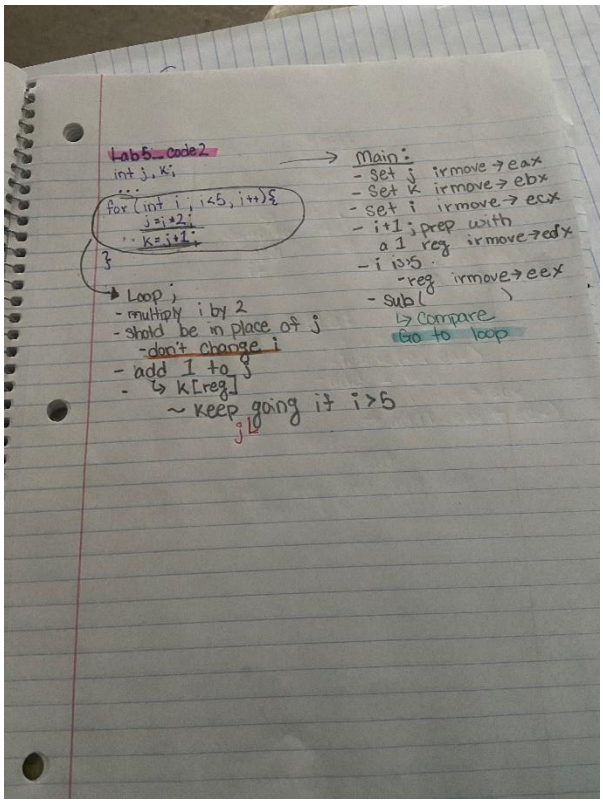

```
0x000:              | Main:
0x000: 30f003000000 |     irmovl $0x3, %eax
0x006: 30f305000000 |     irmovl $0x5, %ebx
0x00c: 6103         |     subl %eax, %ebx
0x00e: 7621000000   |     jg If
0x013:              | Else:
0x013: 30f000000000 |     irmovl $0x0, %eax
0x019: 30f101000000 |     irmovl $0x1, %ecx
0x01f: 6031         |     addl %ebx, %ecx
0x021:              | If:
0x021: 30f105000000 |     irmovl $0x5, %ecx
```

0x027: 6001 | addl %eax, %ecx

0x029: | End:

0x029: 00 | halt

Problem 2:



0x000: | Main:

0x000: 30f004000000 | irmovl $0x4, %eax

0x006: 30f302000000 | irmovl $0x2, %ebx

0x00c: 30f100000000 | irmovl $0x0, %ecx

0x012: 30f705000000 | irmovl $0x5, %edi

0x018: 30f201000000 | irmovl $0x1, %edx

0x01e: 6171 | subl %edi, %ecx

0x020: 7425000000 | jne Loop

0x025: | Loop:

0x025: 6001 | addl %eax, %ecx(%ecx)

```
0x027: 6001       |    addl %eax, %ecx(%ecx)
0x029: 6012       |    addl %ecx, %edx
0x02b: 6171       |    subl %edi, %ecx
0x02d: 7225000000 |    jl Loop
0x032:            | End:
0x032: 00         |    halt
```

Problem 3: Code 1

```
        .file   "lab5_prob3_1.c"
        .text
        .section .rodata  //This creates a READ only
.LC0:
        .string  "Hello, world"
        .text
        .globl   main
        .type    main, @function
main:
.LFB0:
        .cfi_startproc //Used at the start of each functions
        endbr64
        pushq   %rbp //pushes the register on the top of the stack
        .cfi_def_cfa_offset 16 //changes the pointer, it is now offset by 16 bytes of the current one
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6 //modifies the location of CFA register 6 will be the new location, the
offset will remain
        subq    $16, %rsp //immediate operand, marked with a long (16)
        movl    %edi, -4(%rbp) //indexing is done by the register, moves the contents from the offset to
the cell pointed at rbp to edi
```

```
        movq    %rsi, -16(%rbp)
        leaq    .LC0(%rip), %rdi
        call    puts@PLT //most of the module can continue to be mapped to the module instead of swap
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8 //takes register address and adds the offset, defines a rule for computing CFA
        ret
        .cfi_endproc //end of the function
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

Problem 3: Code 2

```
        .file   "lab5_prob3_2.c"
        .text
        .section .rodata
.LC0:
        .string "The value of  i is %d\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc // starts the program
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $32, %rsp
        movl    %edi, -20(%rbp)
        movq    %rsi, -32(%rbp)
        movl    $1, -4(%rbp)
        addl    $1, -4(%rbp) //the values are added together and stored in rbp. Using parathesis indicates indirect memory addressing thus the register is treated as more of a pointer.
        movl    -4(%rbp), %eax
        movl    %eax, %esi
        leaq    .LC0(%rip), %rdi
```

```
        movl    $0, %eax
        call    printf@PLT
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc //ends the program
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

Problem 4:

```
        .file   "lab5_prob4_main.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc  //starts the program
        endbr64
        pushq   %rbp //pushes the register onto the top of the stack
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6 //changes the pointer, it is now offset by 16 bytes of the current one
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movq    %rsi, -16(%rbp)
        movl    $0, %eax
        call    print_hello //enters the print hello function
        movl    $0, %eax
        leave // leaves the call to the function of print hello
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc //ends the program
.LFE0:
        .size   main, .-main
        .section .rodata
.LC0:
        .string "Hello, world"
        .text
```

```
        .globl  print_hello
        .type   print_hello, @function
print_hello: //creates the function to print hello
.LFB1:
        .cfi_startproc //starts the program
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rdi
        call    puts@PLT
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc //ends the program
.LFE1:
        .size   print_hello, .-print_hello
        .ident  "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
```

```
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

Problem four is going to compile the main program than go into the compilation of the function and then travel back to the rest of the main but uses the idea of jumps to go back and forth between the function.

Problem 5 Main:

```
        .file   "lab5_prob5_main.c"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        subq    $16, %rsp
        movl    %edi, -4(%rbp)
        movq    %rsi, -16(%rbp)
```

```
        movl    $0, %eax
        call    print_hello@PLT
        movl    $0, %eax
        leave
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size   main, .-main
        .ident  "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

Problem 5 print:

```
        .file   "lab5_prob5_print.c"
```

```
        .text
        .section .rodata
.LC0:
        .string  "Hello, world"
        .text
        .globl   print_hello
        .type    print_hello, @function
print_hello:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
        .cfi_offset 6, -16
        movq    %rsp, %rbp
        .cfi_def_cfa_register 6
        leaq    .LC0(%rip), %rdi
        call    puts@PLT
        nop
        popq    %rbp
        .cfi_def_cfa 7, 8
        ret
        .cfi_endproc
.LFE0:
        .size    print_hello, .-print_hello
        .ident   "GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0"
        .section .note.GNU-stack,"",@progbits
        .section .note.gnu.property,"a"
        .align 8
```

```
        .long   1f - 0f
        .long   4f - 1f
        .long   5
0:
        .string "GNU"
1:
        .align 8
        .long   0xc0000002
        .long   3f - 2f
2:
        .long   0x3
3:
        .align 8
4:
```

When the lab is put together in assembly code there is not much difference if the codes are put into the same file or in other files, this is because to the machine these are the same because the machine just calls the code and compiles it into something else but to the machine these look just the same as the code.