Anthony Chavez

EEE 174-CpE 185 Summer 2020

Monday, Wednesday

Lab 1

X86 and C Refresher Lab

Dennis Dahlquist

## Introduction:

The purpose of this lab is to become familiar with how microprocessors work. This includes programming in machine, assembly, and C language, using the debugging tools and techniques, and assemblers in programming microprocessors. In addition, understanding how to hand assemble instructions, implement the program development cycle, developing a program from a flow chart, and writing documentation for code will be explored.

## Part 1: Introduction to Debug and C Refresher

For the first part of this section of the lab, I explored the DEBUG mode in the MS-DOS Prompt. First, to enter the DEBUG mode enter “debug” and a “-“ will indicate the DEBUG mode is running. The “?” is the help command and displays the DEBUG commands and their syntax (see Figure 1.1). Next, the “dump” command (“d”) displays the contents of the memory locations (see Figure 1.2). There are 3 data columns: the left side shows the Code Segment (CS) and the Instruction Pointer (IP), the middle is the data stored in memory, and the right side is the data stored in the middle trying to be converted to a printable character in ASCII. The addresses are in ascending 4 digit hexadecimal order but displayed every 3 decimals and the data segments are 2 digit hexadecimal and there are 16 in each row with the half way point marked by a “-“. Third, the “enter” command (“e”) is used to enter the assembly language program and change the data in the memory (see Figure 1.3). Fourth, the “unassembled” command (“u”) is used to see the program you just entered, and it will be converted to assembly language (see Figure 1.4). Fifth, the “register modify” command (“r”) is used to set the Instruction Pointer (IP) register to point to a given address location (see Figure 1.5). Sixth, the “trace” command (“t”) is used to step through a program you just entered where you can see all the register values and the next instruction to be executed (see Figures 1.6 and 1.8 for tracing command and Figures 1.7 and 1.9 for tracing charts). Finally, the “go” command (“g”) is used to run the code until it reaches a specified address (breakpoint) (see Figure 1.10).

For the second part of this section of the lab, I refreshed my knowledge of the C language. First, I made a “Hello World” C program which simple outputs the message, “Hello World” on the screen (see Figure 1.11). Second, I made a program that prompts the user to input two number to be added and outputs the sum to the screen (see Figure 1.12). Last, I converted the assembly mnemonics to a C program. I stored used three integer variables to represent the AX, BX, and DX register as well as making a one-dimensional array to represent the two memory locations 0200 and 0202 (see Figure 1.13).

For the pre-lab section of part 1, I drew up the flow chart and the hand assembly for the Assembly Program (see Figures 1.14 and 1.16). Also the commented Assembly Program can be seen in Figure 1.15 using the analogy of a water tanking filling program.

```
C:\WINDOWS>DEBUG
-?
assemble      A [address]
compare       C range address
dump          D [range]
enter         E address [list]
fill          F range list
go            G [=address] [addresses]
hex           H value1 value2
input         I port
load          L [address] [drive] [firstsector] [number]
move          M range address
name          N [pathname] [arglist]
output        O port byte
proceed       P [=address] [number]
quit          Q
register      R [register]
search        S range list
trace         T [=address] [value]
unassemble    U [range]
write         W [address] [drive] [firstsector] [number]
allocate expanded memory        XA [#pages]
deallocate expanded memory      XD [handle]
map expanded memory pages       XM [Lpage] [Ppage] [handle]
display expanded memory status  XS
```

Figure 1.1: Entering DEBUG mode and using the help command (“?”)

```
a) d 0100
Result:
-d 0100
0F68:0100  DE E8 45 FA AC AA 3C 0D-75 FA 56 8B 36 92 DE 89   ..E...<.u.V.6...
0F68:0110  4C FE 5E 8E 06 08 D3 26-80 3E 43 04 34 00 57 0F   L.^....&.>C.4.W.
0F68:0120  BA 42 86 E9 65 FE BF 81-00 8B 36 92 DE 8B 44 FE   .B..e.....6...D.
0F68:0130  BE C6 DB 8B 74 09 03 C6-50 E8 0D FA 58 E8 5A 00   ....t...P...X.Z.
0F68:0140  03 F1 2B C6 8B C8 E8 7B-F4 83 F9 7F 72 0B B9 7E   ..+....{....r..~
0F68:0150  00 F3 A4 B0 0D AA 47 EB-08 AC AA 3C 0D 74 02 EB   ......G....<.t..
0F68:0160  F8 8B CF 81 E9 82 00 26-88 0E 80 00 C3 8B 1E 92   .......&........
0F68:0170  DE BE 1A D4 BA FF FF B8-00 AE CD 2F 3C 00 C3 A0   .........../<...

b) d 0100 0110
Result:
-d 0100 0110
0F68:0100  DE E8 45 FA AC AA 3C 0D-75 FA 56 8B 36 92 DE 89   ..E...<.u.V.6...
0F68:0110  4C                                                L

c) d 0100 0200
Result:
-d 0100 0200
0F68:0100  DE E8 45 FA AC AA 3C 0D-75 FA 56 8B 36 92 DE 89   ..E...<.u.V.6...
0F68:0110  4C FE 5E 8E 06 08 D3 26-80 3E 43 04 34 00 57 0F   L.^....&.>C.4.W.
0F68:0120  BA 42 86 E9 65 FE BF 81-00 8B 36 92 DE 8B 44 FE   .B..e.....6...D.
0F68:0130  BE C6 DB 8B 74 09 03 C6-50 E8 0D FA 58 E8 5A 00   ....t...P...X.Z.
0F68:0140  03 F1 2B C6 8B C8 E8 7B-F4 83 F9 7F 72 0B B9 7E   ..+....{....r..~
0F68:0150  00 F3 A4 B0 0D AA 47 EB-08 AC AA 3C 0D 74 02 EB   ......G....<.t..
0F68:0160  F8 8B CF 81 E9 82 00 26-88 0E 80 00 C3 8B 1E 92   .......&........
0F68:0170  DE BE 1A D4 BA FF FF B8-00 AE CD 2F 3C 00 C3 A0   .........../<...
0F68:0180  DB E2 0A C0 74 09 56 57-E8 2A 21 5F 5E 73 0A B9   ....t.VW.*!_^s..
0F68:0190  04 01 FC 56 57 F3 A4 5F-5E C3 50 56 33 C9 33 DB   ...VW.._^.PV3.3.
0F68:01A0  AC E8 5F 23 74 19 3C 0D-74 15 F6 C7 20 75 06 3A   .._#t.<.t... u.:
0F68:01B0  06 0C D3 74 0A 41 3C 22-75 E6 80 F7 20 EB E1 5E   ...t.A<"u... ..^
0F68:01C0  58 C3 A1 E1 D7 8B 36 E3-D7 C6 06 25 D9 00 C6 06   X.....6....%....
0F68:01D0  21 D9 00 8B 36 E3 D7 8B-0E E1 D7 8B D6 E3 42 51   !...6.........BQ
0F68:01E0  56 5B 2B DE 59 03 CB 8B-D6 C6 06 C5 DB 00 E3 31   V[+.Y..........1
0F68:01F0  49 AC E8 D9 F6 74 08 49-46 FE 06 C5 DB EB EF E8   I....t.IF.......
0F68:0200  DB                                                .
```

Figure 1.2: Using the “dump” command (“d”)

```
Using the "e" command several times
-e100
0F68:0100  DE.BA   E8.20   45.01   FA.A1   AC.00   AA.02   3C.8B
0D.1E
-E108
0F68:0108  75.02   FA.02   56.29   8B.D8   36.7D   92.06   DE.01
89.D0
-e0110
0F68:0110  4C.7D   FE.02   5E.EB   8E.FA   06.A3   08.00   D3.02
26.CD
-e0118
0F68:0118  80.20

Using the "e" command once
-e100
0F68:0100  DE.BA   E8.20   45.01   FA.A1   AC.00   AA.02   3C.8B
0D.1E
0F68:0108  75.02   FA.02   56.29   8B.D8   36.7D   92.06   DE.01
89.D0
0F68:0110  4C.7D   FE.02   5E.EB   8E.FA   06.A3   08.00   D3.02
26.CD
0F68:0118  80.20
```

Figure 1.3: Using the “enter” command (“e”)

```
-u100 118
0F68:0100 BA2001        MOV     DX,0120
0F68:0103 A10002        MOV     AX,[0200]
0F68:0106 8B1E0202      MOV     BX,[0202]
0F68:010A 29D8          SUB     AX,BX
0F68:010C 7D06          JGE     0114
0F68:010E 01D0          ADD     AX,DX
0F68:0110 7D02          JGE     0114
0F68:0112 EBFA          JMP     010E
0F68:0114 A30002        MOV     [0200],AX
0F68:0117 CD20          INT     20
```

Figure 1.4: Using the “unassembled” command (“u”)

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000
DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0100   NV UP EI PL NZ NA PO
NC
0F68:0100 BA2001        MOV     DX,0120
```

Figure 1.5: Using the “register modify” command (“r”)

```
-e200
0F68:0200  DB.20   F9.01   75.50   04.02
-d200 203
0F68:0200  20 01 50 02                                          .P.
-t

AX=0000  BX=0000  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0103   NV UP EI PL NZ NA PO NC
0F68:0103 A10002        MOV     AX,[0200]
DS:0200=0120
-t

AX=0120  BX=0000  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0106   NV UP EI PL NZ NA PO NC
0F68:0106 8B1E0202      MOV     BX,[0202]
DS:0202=0250
-t

AX=0120  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010A   NV UP EI PL NZ NA PO NC
0F68:010A 29D8          SUB     AX,BX
-t

AX=FED0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010C   NV UP EI NG NZ NA PO CY
0F68:010C 7D06          JGE     0114
-t

AX=FED0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010E   NV UP EI NG NZ NA PO CY
0F68:010E 01D0          ADD     AX,DX
-t

AX=FFF0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0110   NV UP EI NG NZ NA PE NC
0F68:0110 7D02          JGE     0114
-t

AX=FFF0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0112   NV UP EI NG NZ NA PE NC
0F68:0112 EBFA          JMP     010E
-t

AX=FFF0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010E   NV UP EI NG NZ NA PE NC
0F68:010E 01D0          ADD     AX,DX
-t

AX=0110  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0110   NV UP EI PL NZ NA PO CY
0F68:0110 7D02          JGE     0114
```

```
-t

AX=0110  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0114   NV UP EI PL NZ NA PO CY
0F68:0114 A30002        MOV     [0200],AX
DS:0200=0120
-t

AX=0110  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0117   NV UP EI PL NZ NA PO CY
0F68:0117 CD20          INT     20
-p

Program terminated normally
```

Figure 1.6: Using the “trace” command (“t”)

EEE 174-CpE 185
Laboratory Exercise #1 — Name: Anthony Chavez

Program Tracing Chart

Registers:

| | AX: | BX: | CX: | DX: | OF: | ZF: | SF: | CS: | IP: | DS:200 | DS:202 | Next Instruction |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Value: ---> | 0000 | 0000 | 0000 | 0000 | NV (0) | NZ (0) | PL (0) | 0F68 | 0100 | 0120 | 0250 | MOV DX,0120 |
| | 0000 | 0000 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0103 | 0120 | 0250 | MOV AX,[0200] |
| | 0120 | 0000 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0106 | 0120 | 0250 | MOV BX,[0202] |
| | 0120 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 010A | 0120 | 0250 | SUB AX,BX |
| | FED0 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | NG (1) | 0F68 | 010C | 0120 | 0250 | JGE 0114 |
| | FED0 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | NG (1) | 0F68 | 010E | 0120 | 0250 | ADD AX,DX |
| | FFF0 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | NG (1) | 0F68 | 0110 | 0120 | 0250 | JGE 0114 |
| | FFF0 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | NG (1) | 0F68 | 0112 | 0120 | 0250 | JMP 010E |
| | FFF0 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | NG (1) | 0F68 | 010E | 0120 | 0250 | ADD AX,DX |
| | 0110 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0110 | 0120 | 0250 | JGE 0114 |
| | 0110 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0114 | 0120 | 0250 | MOV [0200],AX |
| | 0110 | 0250 | 0000 | 0120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0117 | 0110 | 0250 | INT 20 |

Figure 1.7: Tracing Chart for first run

```
-E100
0F68:0100  DE.BA   E8.20   45.A1   FA.A1   AC.00   AA.02   3C.8B   0D.1E
0F68:0108  75.02   FA.02   56.29   8B.D8   36.7D   92.06   DE.01   89.D0
0F68:0110  4C.7D   FE.02   5E.EB   8E.FA   06.A3   08.00   D3.02   26.CD
0F68:0118  80.20
-U100 118
0F68:0100 BA20A1        MOV     DX,A120
0F68:0103 A10002        MOV     AX,[0200]
0F68:0106 8B1E0202      MOV     BX,[0202]
0F68:010A 29D8          SUB     AX,BX
0F68:010C 7D06          JGE     0114
0F68:010E 01D0          ADD     AX,DX
0F68:0110 7D02          JGE     0114
0F68:0112 EBFA          JMP     010E
0F68:0114 A30002        MOV     [0200],AX
0F68:0117 CD20          INT     20
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0100   NV UP EI PL NZ NA PO NC
0F68:0100 BA20A1        MOV     DX,A120
-E200
0F68:0200  DB.01   F9.20   75.20   04.50
-D200 203
0F68:0200  01 20 20 50                                          .  P
-T

AX=0000  BX=0000  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0103   NV UP EI PL NZ NA PO NC
0F68:0103 A10002        MOV     AX,[0200]
DS:0200=2001
-T

AX=2001  BX=0000  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0106   NV UP EI PL NZ NA PO NC
0F68:0106 8B1E0202      MOV     BX,[0202]
DS:0202=5020
-T

AX=2001  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010A   NV UP EI PL NZ NA PO NC
0F68:010A 29D8          SUB     AX,BX
-T

AX=CFE1  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010C   NV UP EI NG NZ NA PE CY
0F68:010C 7D06          JGE     0114
-T


AX=CFE1  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010E   NV UP EI NG NZ NA PE CY
0F68:010E 01D0          ADD     AX,DX
```

```
-T

AX=7101  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0110   OV UP EI PL NZ NA PO CY
0F68:0110 7D02          JGE     0114
-T

AX=7101  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0112   OV UP EI PL NZ NA PO CY
0F68:0112 EBFA          JMP     010E
-T

AX=7101  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010E   OV UP EI PL NZ NA PO CY
0F68:010E 01D0          ADD     AX,DX
-T

AX=1221  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0110   NV UP EI PL NZ NA PE CY
0F68:0110 7D02          JGE     0114
-T

AX=1221  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0114   NV UP EI PL NZ NA PE CY
0F68:0114 A30002        MOV     [0200],AX
DS:0200=2001
-T

AX=1221  BX=5020  CX=0000  DX=A120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0117   NV UP EI PL NZ NA PE CY
0F68:0117 CD20          INT     20
-P

Program terminated normally
```

Figure 1.8: Tracing for second run

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EEE 174-CpE 185 | | | | | | | | | | | | |
| Laboratory Exercise #1 | | | | | | | | | Name: Anthony Chavez | | | |
| | | | | | | | | | | | | |
| Program Tracing Chart | | | | | | | | | | | | |
| | | | Registers: | | | | | | | | | |
| | | | | | | | | | | | | |
| | AX: | BX: | CX: | DX: | OF: | ZF: | SF: | CS: | IP: | DS:200 | DS:202 | Next Instruction |
| Value: ---> | 0000 | 0000 | 0000 | 0000 | NV (0) | NZ (0) | PL (0) | 0F68 | 0100 | 2001 | 5020 | MOV DX,A120 |
| | 0000 | 0000 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0103 | 2001 | 5020 | MOV AX,[0200] |
| | 2001 | 0000 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0106 | 2001 | 5020 | MOV BX,[0202] |
| | 2001 | 5020 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 010A | 2001 | 5020 | SUB AX,BX |
| | CFE1 | 5020 | 0000 | A120 | NV (0) | NZ (0) | NG (1) | 0F68 | 010C | 2001 | 5020 | JGE 0114 |
| | CFE1 | 5020 | 0000 | A120 | NV (0) | NZ (0) | NG (1) | 0F68 | 010E | 2001 | 5020 | ADD AX,DX |
| | 7101 | 5020 | 0000 | A120 | OV (1) | NZ (0) | PL (0) | 0F68 | 0110 | 2001 | 5020 | JGE 0114 |
| | 7101 | 5020 | 0000 | A120 | OV (1) | NZ (0) | PL (0) | 0F68 | 0112 | 2001 | 5020 | JMP 010E |
| | 7101 | 5020 | 0000 | A120 | OV (1) | NZ (0) | PL (0) | 0F68 | 010E | 2001 | 5020 | ADD AX,DX |
| | 1221 | 5020 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0110 | 2001 | 5020 | JGE 0114 |
| | 1221 | 5020 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0114 | 2001 | 5020 | MOV [0200],AX |
| | 1221 | 5020 | 0000 | A120 | NV (0) | NZ (0) | PL (0) | 0F68 | 0117 | 1221 | 5020 | INT 20 |

Figure 1.9: Tracing chart for second run

```
-E100
0F68:0100  DE.BA   E8.20   45.01   FA.A1   AC.00   AA.02   3C.8B
0D.1E
0F68:0108  75.02   FA.02   56.29   8B.D8   36.7D   92.06   DE.01
89.D0
0F68:0110  4C.7D   FE.02   5E.EB   8E.FA   06.A3   08.00   D3.02
26.CD
0F68:0118  80.20
-E200
0F68:0200  DB.20   F9.01   75.50   04.02
-U100 118
0F68:0100 BA2001        MOV     DX,0120
0F68:0103 A10002        MOV     AX,[0200]
0F68:0106 8B1E0202      MOV     BX,[0202]
0F68:010A 29D8          SUB     AX,BX
0F68:010C 7D06          JGE     0114
0F68:010E 01D0          ADD     AX,DX
0F68:0110 7D02          JGE     0114
0F68:0112 EBFA          JMP     010E
0F68:0114 A30002        MOV     [0200],AX
0F68:0117 CD20          INT     20
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0100   NV UP EI PL NZ NA PO NC
0F68:0100 BA2001        MOV     DX,0120
-G=100 10E

AX=FED0  BX=0250  CX=0000  DX=0120  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010E   NV UP EI NG NZ NA PO CY
0F68:010E 01D0          ADD     AX,DX
```

Figure 1.10: Using the “go” command (“g”)

```c
/*
 This programs outputs the message "Hello World"
 */
#include <stdio.h>

int main()
{
    printf("Hello World");
    return 0;
}
```

```
 ----jGRASP exec: C:\Users\[redacted]
Hello World
 ----jGRASP: operation complete.
```

Figure 1.11: “Hello World” program in C

```c
/*
 This programs asks the user to input two
 whole numbers, adds them together and
 outputs the sum.
 */
#include<stdio.h>

int main() {
    int a, b, c;

    printf("Enter the first number: ");
    scanf("%d", &a);
    printf("Enter the second number: ");
    scanf("%d", &b);

    c = a + b;

    printf("Sum of the numbers = %d\n", c);

    return 0;
}
```

```
 ----jGRASP exec: C:\Users\[redacted]
Enter the first number: 1
Enter the second number: 1
Sum of the numbers = 2

 ----jGRASP: operation complete.
```

Figure 1.12: Add two numbers in C

```c
/*
 This programs performs the same function as the
 assembly language program used in DEBUG.
 */
#include<stdio.h>

int main() {
    int mem[2] = {120, 250};    // Memory Addresses 0200 and 0202

    int D = 120,                // Register DX ; MOV DX,##
        A = mem[0],             // Register AX ; MOV AX,[##]
        B = mem[1];             // Register BX ; MOV BX,[##]

    printf("Memory 0200 is: %d\n", mem[0]);
    printf("Memory 0202 is: %d\n", mem[1]);
    printf("Value of DX register: %d\n", D);
    printf("Value in AX register: %d\n", A);
    printf("Value in BX register: %d\n", B);

    A = A - B;                  // SUB AX,BX

    printf("Value in AX register: %d\n", A);

    if(A < 0) {
        while(A < 0) {              // JGE 0114
            A = A + D;              // ADD AX,DX
            printf("Value in AX register: %d\n", A);
        }
    }

    mem[0] = A;                 // MOV [0200],AX

    printf("Value in Memory 0200: %d\n", mem[0]);

    return 0;                   // INT 20
}
```

```
 ----jGRASP exec: C:\Users\[redacted]
Memory 0200 is: 120
Memory 0202 is: 250
Value of DX register: 120
Value in AX register: 120
Value in BX register: 250
Value in AX register: -130
Value in AX register: -10
Value in AX register: 110
Value in Memory 0200: 110

 ----jGRASP: operation complete.
```

Figure 1.13: Assembly Language Program Converted to C Program

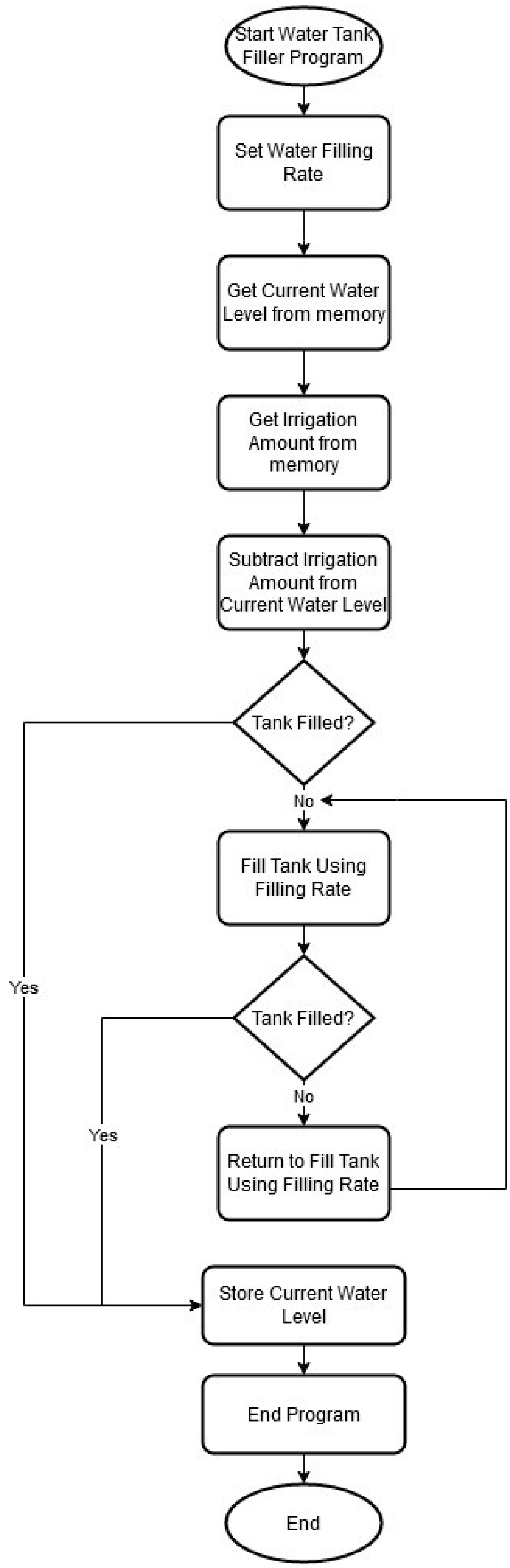



Figure 1.14: Flow Chart of the Assembly Program

```
MOV     DX,0120              ; Set Water Filling Rate
MOV     AX,[0200]            ; Get Current Water Level from memory
MOV     BX,[0202]            ; Get Irrigation Amount from memory
SUB     AX,BX                ; Subtract Irrigation Amount from Current Water Level
JGE     0114                 ; Tank Filled?
ADD     AX,DX                ; Fill Tank using Filling Rate
JGE     0114                 ; Tank Filled?
JMP     010E                 ; Return to Fill Tank using Filling Rate
MOV     [0200],AX            ; Store Current Water Level
INT     20                   ; End Program
```

Figure 1.15: Commented Assembly Program Using Flow Chart Application

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **EEE 174-CpE 185** | | | | | | | | | |
| **Laboratory Hand-Assembly** | | | | | | | | | |
| **Dahlquist/Stoffers/Schultz** | | | | | | | | | |
| Instruction: | MOV DX, 0120 | | | | | | | | |
| Address: | CS | **: 0100** | | Operation: | MOV | Dest.: | DX | Source: | **120** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 1 | reg = 010 | immediate data = 2001h | | | | | |
| Binary: | 1011 | 1010 | 2001h | | | | | | |
| | B | A | 2001 | | | | | | |
| Hex: | BA2001 | | | | | | | | |
| Instruction: | MOV AX, [0200] | | | | | | | | |
| Address: | CS | **: 0103** | | Operation: | MOV | Dest.: | AX | Source: | **200** |
| | | memory to AX | | | | | | | |
| Instruction Format | | 1010 000w full displacement | | | | | | | |
| | | w = 1 | | | | | | | |
| Binary: | 1010 | 0001 | 0002h | | | | | | |
| | A | 1 | 2 | | | | | | |
| Hex: | A10002 | | | | | | | | |
| Instruction: | MOV BX,[0202] | | | | | | | | |
| Address: | CS | **: 0106** | | Operation: | MOV | Dest.: | BX | Source: | **202** |
| | | memory to reg | | | | | | | |
| Instruction Format | | 1000 101w mod reg r/m | | | | | | | |
| | | w = 1 | mod = 00 | reg = 011 | r/m = 110 | | | | |
| Binary: | 1000 | 1011 | 0001 | 1110 | 0202h | | | | |
| | 8 | B | 1 | E | 0202h | | | | |
| Hex: | 8B1E0202 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | SUB AX,BX | | | | | | | | |
| Address: | CS | : **010A** | | Operation: | SUB | Dest.: | AX | Source: | **BX** |
| Instruction Format | | register1 to register2 | | | | | | | |
| | | w = 1 | reg1 = 011 | reg2 = 000 | | | | | |
| Binary: | 0010 | 1001 | 1101 | 1000 | | | | | |
| | 2 | 9 | D | 8 | | | | | |
| Hex: | 29D8 | | | | | | | | |
| Instruction: | JGE 0114 | | | | | | | | |
| Address: | CS | : **010C** | | Operation: | JGE | Dest.: | 114 | Source: | |
| Instruction Format | | 0111 tttn 8-bit displacement | | | | | | | |
| | | tttn = 1101 | | | | | | | |
| Binary: | 0111 | 1101 | | | | | | | |
| | 7 | D | 06h | | | | | | |
| Hex: | 7D06 | | | | | | | | |
| Instruction: | ADD AX,DX | | | | | | | | |
| Address: | CS | : **010E** | | Operation: | ADD | Dest.: | AX | Source: | **DX** |
| Instruction Format | | 0000 000w 11reg1 reg2 | | | | | | | |
| | | w = 1 | reg1 = 010 | reg2 = 000 | | | | | |
| Binary: | 0000 | 0001 | 1101 | 0000 | | | | | |
| | 0 | 1 | D | 0 | | | | | |
| Hex: | 01D0 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | JGE 0114 | | | | | | | | |
| Address: | CS | **: 0110** | | Operation: | JGE | Dest.: | 114 | Source: | |
| Instruction Format | | 0111 tttn 8-bit displacement | | | | | | | |
| | | tttn = 1101 | | | | | | | |
| Binary: | 0111 | 1101 | | | | | | | |
| | 7 | D | 02h | | | | | | |
| Hex: | 7D02 | | | | | | | | |
| Instruction: | JMP 010E | | | | | | | | |
| Address: | CS | **: 0112** | | Operation: | JMP | Dest.: | 010E | Source: | |
| Instruction Format | | 1110 1011 8-bit displacement | | | | | | | |
| Binary: | 1110 | 1011 | | | | | | | |
| | E | D | FA | | | | | | |
| Hex: | EDFA | | | | | | | | |
| Instruction: | MOV [0200],AX | | | | | | | | |
| Address: | CS | **: 0114** | | Operation: | MOV | Dest.: | 200 | Source: | **AX** |
| Instruction Format | | 1010 001w full displacment | | | | | | | |
| | | w = 1 | | | | | | | |
| Binary: | 1010 | 0011 | 0002h | | | | | | |
| | A | 3 | 0002h | | | | | | |
| Hex: | A30002 | | | | | | | | |
| Instruction: | INT 20 | | | | | | | | |
| Address: | CS | **: 0117** | | Operation: | INT | Dest.: | 20 | Source: | |
| | | INT n – Interrupt Type n | | | | | | | |
| Instruction Format | | 1100 1101 : type | | | | | | | |
| | | n=20 or 0010 0000 | | | | | | | |
| Binary: | 1100 | 1101 | 0010 | 0000 | | | | | |
| | C | D | 2 | 0 | | | | | |
| Hex: | CD20 | | | | | | | | |

Figure 1.16: Hand Assembly of Assembly Program

## Part 2: Hand Assembly and C Programming

For this section of the lab, I reprogrammed the original Assembly Program in part 1 which used the 16-bit version of the registers to only use the 8-bit version of the registers. In addition, I had to account for only using one conditional jump (JGE), not using the specified register, using a given starting memory location, displaying a title for the program, and adding a counter to keep track of how many times the program ran in the loop.

First, I determined the register I was not allowed to use by looking up my last name in the RegMemLocation pdf and found I was not allowed to use the AX register. Second, I modified the flow chart from part 1 to only have only one conditional jump (see Figure 2.1). Third, I hand-assembled the instructions for the program and debugged the program in the debugger (see Figure 2.2). Fourth, I modified the flow chart again to accommodate the program title and loop counter (see Figure 2.3) and hand-assembled the instructions (see Figure 2.4). Then, I ran two traces through my full program (see Figures 2.5 – 2.10). Finally, I converted the full program to C and tried to convert the program to Inline Assembly, but I found it too difficult to understand (see Figure 2.12).

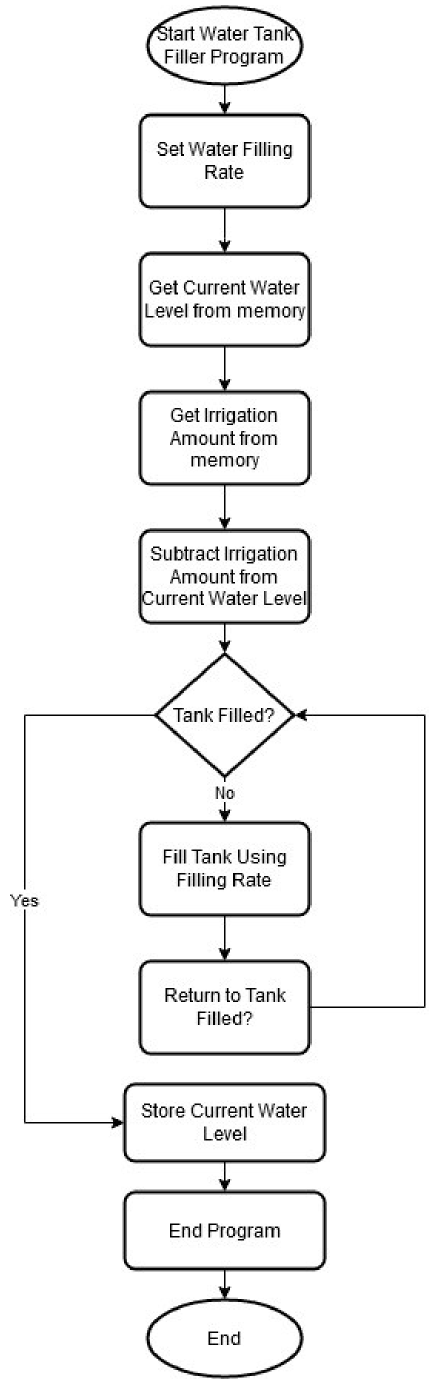



Figure 2.1: Flow chart of Program without title and name

**EEE 174-CpE 185**
**Laboratory Hand-Assembly**
**Dahlquist/Stoffers/Schultz**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV DL, 20 | | | | | | | | |
| Address: | CS | **: 0100** | | Operation: | MOV | Dest.: | DL | Source: | **20** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 0 | reg = 010 | immediate data = 20h | | | | | |
| Binary: | 1011 | 0010 | 20h | | | | | | |
| | B | 2 | 20h | | | | | | |
| Hex: | B220 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV BL, [0204] | | | | | | | | |
| Address: | CS | **: 0103** | | Operation: | MOV | Dest.: | BL | Source: | **204** |
| | | memory to reg | | | | | | | |
| Instruction Format | | 1000 101w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 011 | r/m = 110 | | | | |
| Binary: | 1000 | 1010 | 0001 | 1110 | 0204h | | | | |
| | 8 | A | 1 | E | 0204h | | | | |
| Hex: | 8A1E0402 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV CL,[0206] | | | | | | | | |
| Address: | CS | **: 0106** | | Operation: | MOV | Dest.: | CL | Source: | **206** |
| | | memory to reg | | | | | | | |
| Instruction Format | | 1000 101w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 001 | r/m = 110 | | | | |
| Binary: | 1000 | 1010 | 0000 | 1110 | 0206h | | | | |
| | 8 | A | 0 | E | 0205h | | | | |
| Hex: | 8A0E0502 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | SUB BL,CL | | | | | | | | |
| Address: | CS | : 010A | | Operation: | SUB | Dest.: | BL | Source: | CL |
| Instruction Format | | 0010 101w 11register1 to register2 | | | | | | | |
| | | w = 0 | reg1 = 011 | reg2 = 001 | | | | | |
| Binary: | 0010 | 1010 | 1101 | 1001 | | | | | |
| | 2 | A | D | 9 | | | | | |
| Hex: | 2AD9 | | | | | | | | |
| Instruction: | JGE 0112 | | | | | | | | |
| Address: | CS | : 010C | | Operation: | JGE | Dest.: | 112 | Source: | |
| Instruction Format | | 0111 tttn 8-bit displacement | | | | | | | |
| | | tttn = 1101 | | | | | | | |
| Binary: | 0111 | 1101 | | | | | | | |
| | 7 | D | 04h | | | | | | |
| Hex: | 7D04 | | | | | | | | |
| Instruction: | ADD BL,DL | | | | | | | | |
| Address: | CS | : 010E | | Operation: | ADD | Dest.: | BL | Source: | DL |
| Instruction Format | | 0000 001w 11reg1 reg2 | | | | | | | |
| | | w = 0 | reg1 = 011 | reg2 = 010 | | | | | |
| Binary: | 0000 | 0010 | 1101 | 1010 | | | | | |
| | 0 | 2 | D | A | | | | | |
| Hex: | 02DA | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | JMP 010C | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0110** | | Operation: | JMP | Dest.: | 010C | Source: | |
| | | | | | | | | | |
| Instruction Format | | 1110 1011 8-bit displacement | | | | | | | |
| | | | | | | | | | |
| Binary: | 1110 | 1011 | | | | | | | |
| | E | B | FA | | | | | | |
| Hex: | EBFA | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV [0200],BL | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0114** | | Operation: | MOV | Dest.: | 204 | Source: | **BL** |
| | | | | | | | | | |
| Instruction Format | | 1000 100w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 011 | r/m = 110 | | | | |
| Binary: | 1000 | 1000 | 0001 | 1110 | | | | | |
| | 8 | 8 | 1 | E | | | | | |
| Hex: | 881E0402 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | INT 20 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0117** | | Operation: | INT | Dest.: | 20 | Source: | |
| | | INT n – Interrupt Type n | | | | | | | |
| Instruction Format | | 1100 1101 : type | | | | | | | |
| | | n=20 or 0010 0000 | | | | | | | |
| Binary: | 1100 | 1101 | 0010 | 0000 | | | | | |
| | C | D | 2 | 0 | | | | | |
| Hex: | CD20 | | | | | | | | |

Figure 2.2: Hand Assembly of Program without title and name

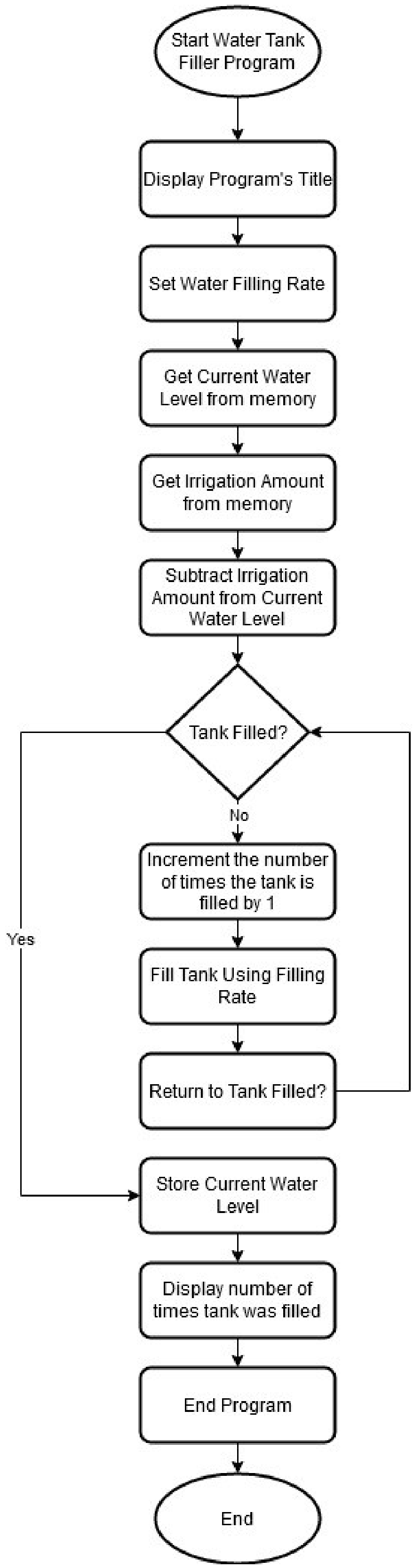



Figure 2.3: Flow chart of Full Program

**EEE 174-CpE 185**
**Laboratory Hand-Assembly**
**Dahlquist/Stoffers/Schultz**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV DX, 210 | | | | | | | | |
| Address: | CS | **: 0100** | | Operation: | MOV | Dest.: | DX | Source: | **210** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 1 | reg = 010 | immediate data = 210h | | | | | |
| Binary: | 1011 | 1010 | 210h | | | | | | |
| | B | A | 210h | | | | | | |
| Hex: | BA1002 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV AH, 09 | | | | | | | | |
| Address: | CS | **: 0103** | | Operation: | MOV | Dest.: | AH | Source: | **09** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 0 | reg = 100 | immediate data = 09h | | | | | |
| Binary: | 1011 | 0100 | 09h | | | | | | |
| | B | 4 | 09h | | | | | | |
| Hex: | B409 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | INT 21 | | | | | | | | |
| Address: | CS | **: 0105** | | Operation: | INT | Dest.: | 21 | Source: | |
| | | INT n – Interrupt Type n | | | | | | | |
| Instruction Format | | 1100 1101 : type | | | | | | | |
| | | n=21 or 0010 0001 | | | | | | | |
| Binary: | 1011 | 0100 | 0010 | 0001 | | | | | |
| | C | D | 2 | 1 | | | | | |
| Hex: | CD21 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV DL, 20 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0107** | | Operation: | MOV | Dest.: | DL | Source: | **20** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 0 | reg = 010 | immediate data = 20h | | | | | |
| Binary: | 1011 | 0010 | 20h | | | | | | |
| | B | 2 | 20h | | | | | | |
| Hex: | B220 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV BL, [0204] | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0109** | | Operation: | MOV | Dest.: | BL | Source: | **204** |
| | | memory to reg | | | | | | | |
| Instruction Format | | 1000 101w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 011 | r/m = 110 | | | | |
| Binary: | 1000 | 1010 | 0001 | 1110 | 0204h | | | | |
| | 8 | A | 1 | E | 0204h | | | | |
| Hex: | 8A1E0402 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV CL,[0206] | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 010D** | | Operation: | MOV | Dest.: | CL | Source: | **206** |
| | | memory to reg | | | | | | | |
| Instruction Format | | 1000 101w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 001 | r/m = 110 | | | | |
| Binary: | 1000 | 1010 | 0000 | 1110 | 0206h | | | | |
| | 8 | A | 0 | E | 0205h | | | | |
| Hex: | 8A0E0502 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | SUB BL,CL | | | | | | | | |
| Address: | CS | **: 0111** | | Operation: | SUB | Dest.: | BL | Source: | **CL** |
| Instruction Format | | 0010 101w 11register1 to register2 | | | | | | | |
| | | w = 0 | reg1 = 011 | reg2 = 001 | | | | | |
| Binary: | 0010 | 1010 | 1101 | 1001 | | | | | |
| | 2 | A | D | 9 | | | | | |
| Hex: | 2AD9 | | | | | | | | |
| Instruction: | JGE 11D | | | | | | | | |
| Address: | CS | **: 0113** | | Operation: | JGE | Dest.: | 11D | Source: | |
| Instruction Format | | 0111 tttn 8-bit displacement | | | | | | | |
| | | tttn = 1101 | | | | | | | |
| Binary: | 0111 | 1101 | | | | | | | |
| | 7 | D | 08h | | | | | | |
| Hex: | 7D08 | | | | | | | | |
| Instruction: | INC BYTE PTR [0206] | | | | | | | | |
| Address: | CS | **: 0115** | | Operation: | INC | Dest.: | 206 | Source: | |
| Instruction Format | | 1111 111w mod 000 r/m | | | | | | | |
| | | w = 0 | mod = 00 | r/m = 110 | | | | | |
| Binary: | 1111 | 1110 | 0000 | 0110 | 0620h | | | | |
| | F | E | 0 | 6 | 0602h | | | | |
| Hex: | FE060602 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | ADD BL,DL | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | : **0119** | | Operation: | ADD | Dest.: | BL | Source: | **DL** |
| | | | | | | | | | |
| Instruction Format | | 0000 000w 11reg1 reg2 | | | | | | | |
| | | w = 0 | reg1 = 011 | reg2 = 010 | | | | | |
| Binary: | 0000 | 0010 | 1101 | 1010 | | | | | |
| | 0 | 2 | D | A | | | | | |
| Hex: | 02DA | | | | | | | | |
| | | | | | | | | | |
| Instruction: | JMP 112 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | : **011B** | | Operation: | JMP | Dest.: | 112 | Source: | |
| | | | | | | | | | |
| Instruction Format | | 1110 1011 8-bit displacement | | | | | | | |
| | | | | | | | | | |
| Binary: | 1110 | 1011 | | | | | | | |
| | E | B | F6 | | | | | | |
| Hex: | EBF6 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV [0204],BL | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | : **011D** | | Operation: | MOV | Dest.: | 204 | Source: | **BL** |
| | | | | | | | | | |
| Instruction Format | | 1000 100w mod reg r/m | | | | | | | |
| | | w = 0 | mod = 00 | reg = 011 | r/m = 110 | | | | |
| Binary: | 1000 | 1000 | 0001 | 1110 | | | | | |
| | 8 | 8 | 1 | E | 204h | | | | |
| Hex: | 881E0402 | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Instruction: | MOV DX, 0206 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0121** | | Operation: | MOV | Dest.: | DX | Source: | **206** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 1 | reg = 010 | 0602h | | | | | |
| Binary: | 1011 | 1010 | 0620h | | | | | | |
| | B | A | 0620h | | | | | | |
| Hex: | BA0602 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | MOV AH, 09 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0124** | | Operation: | MOV | Dest.: | AH | Source: | **09** |
| | | immediate to register (alternate encoding) | | | | | | | |
| Instruction Format | | 1011 wreg immediate data | | | | | | | |
| | | w = 0 | reg = 100 | immediate data = 09h | | | | | |
| Binary: | 1011 | 0100 | 09h | | | | | | |
| | B | 4 | 09h | | | | | | |
| Hex: | B409 | | | | | | | | |
| | | | | | | | | | |
| Instruction: | INT 21 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0126** | | Operation: | INT | Dest.: | 21 | Source: | |
| | | INT n – Interrupt Type n | | | | | | | |
| Instruction Format | | 1100 1101 : type | | | | | | | |
| | | n=21 or 0010 0001 | | | | | | | |
| Binary: | 1011 | 0100 | 0010 | 0001 | | | | | |
| | C | D | 2 | 1 | | | | | |
| Hex: | CD21 | | | | | | | | |
| Instruction: | INT 20 | | | | | | | | |
| | | | | | | | | | |
| Address: | CS | **: 0128** | | Operation: | INT | Dest.: | 20 | Source: | |
| | | INT n – Interrupt Type n | | | | | | | |
| Instruction Format | | 1100 1101 : type | | | | | | | |
| | | n=20 or 0010 0000 | | | | | | | |
| Binary: | 1100 | 1101 | 0010 | 0000 | | | | | |
| | C | D | 2 | 0 | | | | | |
| Hex: | CD20 | | | | | | | | |

Figure 2.4: Hand Assembly of Full Program

```
-e100
0F68:0100  DE.BA   E8.10   45.02   FA.B4   AC.09   AA.CD   3C.21
0D.B2
0F68:0108  75.20   FA.8A   56.1E   8B.04   36.02   92.8A   DE.0E
89.05
0F68:0110  4C.02   FE.2A   5E.D9   8E.7D   06.08   08.FE   D3.06
26.06
0F68:0118  80.02   3E.02   43.DA   04.EB   34.F6   00.88   57.1E
0F.04
0F68:0120  BA.02   42.BA   86.06   E9.02   65.B4   FE.09   BF.CD
81.21
0F68:0128  00.CD   8B.20
-e204
0F68:0204  FE.F8   06.05
-e210 "Water Filling Program by Anthony Chavez" 0d 0a "$"
-e206 30 0d 0a "$"
-u100 128
0F68:0100 BA1002        MOV     DX,0210
0F68:0103 B409          MOV     AH,09
0F68:0105 CD21          INT     21
0F68:0107 B220          MOV     DL,20
0F68:0109 8A1E0402      MOV     BL,[0204]
0F68:010D 8A0E0502      MOV     CL,[0205]
0F68:0111 2AD9          SUB     BL,CL
0F68:0113 7D08          JGE     011D
0F68:0115 FE060602      INC     BYTE PTR [0206]
0F68:0119 02DA          ADD     BL,DL
0F68:011B EBF6          JMP     0113
0F68:011D 881E0402      MOV     [0204],BL
0F68:0121 BA0602        MOV     DX,0206
0F68:0124 B409          MOV     AH,09
0F68:0126 CD21          INT     21
0F68:0128 CD20          INT     20
```

Figure 2.5: Full Program first run

```
-g=100 107
Water Filling Program by Anthony Chavez

AX=0924  BX=0000  CX=0000  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0107   NV UP EI PL NZ NA PO NC
0F68:0107 B220          MOV     DL,20
-t

AX=0924  BX=0000  CX=0000  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0109   NV UP EI PL NZ NA PO NC
0F68:0109 8A1E0402      MOV     BL,[0204]
DS:0204=F8
-t

AX=0924  BX=00F8  CX=0000  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010D   NV UP EI PL NZ NA PO NC
0F68:010D 8A0E0502      MOV     CL,[0205]
DS:0205=05
-t

AX=0924  BX=00F8  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0111   NV UP EI PL NZ NA PO NC
0F68:0111 2AD9          SUB     BL,CL
-t

AX=0924  BX=00F3  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0113   NV UP EI NG NZ NA PE NC
0F68:0113 7D08          JGE     011D
-t

AX=0924  BX=00F3  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0115   NV UP EI NG NZ NA PE NC
0F68:0115 FE060602      INC     BYTE PTR [0206]
DS:0206=30
-t

AX=0924  BX=00F3  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0119   NV UP EI PL NZ NA PO NC
0F68:0119 02DA          ADD     BL,DL
-t

AX=0924  BX=0013  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=011B   NV UP EI PL NZ NA PO CY
0F68:011B EBF6          JMP     0113
-t

AX=0924  BX=0013  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0113   NV UP EI PL NZ NA PO CY
0F68:0113 7D08          JGE     011D
```

```
-t

AX=0924  BX=0013  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=011D   NV UP EI PL NZ NA PO CY
0F68:011D 881E0402      MOV     [0204],BL
DS:0204=F8
-t

AX=0924  BX=0013  CX=0005  DX=0220  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0121   NV UP EI PL NZ NA PO CY
0F68:0121 BA0602        MOV     DX,0206
-t

AX=0924  BX=0013  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0124   NV UP EI PL NZ NA PO CY
0F68:0124 B409          MOV     AH,09
-t

AX=0924  BX=0013  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0126   NV UP EI PL NZ NA PO CY
0F68:0126 CD21          INT     21
-p
1

AX=0924  BX=0013  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0128   NV UP EI PL NZ NA PO CY
0F68:0128 CD20          INT     20
-p

Program terminated normally
```

Figure 2.6: Full Program first run trace

| EEE 174-CpE 185 | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Laboratory Exercise #2 | | | | | | | | | Name: Anthony Chavez | | | | |
| | | | | | | | | | | | | | |
| Program Tracing Chart | | | | | | | | | | | | | |
| | | | Registers: | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | AX: | BX: | CX: | DX: | OF: | ZF: | SF: | CS: | IP: | DS:204 | DS:205 | DS: 206 | Next Instruction |
| Value: ---> | 0924 | 0000 | 0000 | 0210 | NV (0) | NZ (0) | PL (0) | 0F68 | 0107 | F8 | 05 | 30 | MOV DL,20 |
| | 0924 | 0000 | 0000 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 0109 | F8 | 05 | 30 | MOV BL,[0204] |
| | 0924 | 00F8 | 0000 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 010D | F8 | 05 | 30 | MOV CL,[0205] |
| | 0924 | 00F8 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 0111 | F8 | 05 | 30 | SUB BL,CL |
| | 0924 | 00F3 | 0005 | 0220 | NV (0) | NZ (0) | NG (1) | 0F68 | 0113 | F8 | 05 | 30 | JGE 011D |
| | 0924 | 00F3 | 0005 | 0220 | NV (0) | NZ (0) | NG (1) | 0F68 | 0115 | F8 | 05 | 30 | INC BYTE PTR [0206] |
| | 0924 | 00F3 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 0119 | F8 | 05 | 31 | ADD BL,DL |
| | 0924 | 0013 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 011B | F8 | 05 | 31 | JMP 0113 |
| | 0924 | 0013 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 0113 | F8 | 05 | 31 | JGE 011D |
| | 0924 | 0013 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 011D | F8 | 05 | 31 | MOV [0204],BL |
| | 0924 | 0013 | 0005 | 0220 | NV (0) | NZ (0) | PL (0) | 0F68 | 0121 | 13 | 05 | 31 | MOV DX,0206 |
| | 0924 | 0013 | 0005 | 0206 | NV (0) | NZ (0) | PL (0) | 0F68 | 0124 | 13 | 05 | 31 | MOV AH,09 |
| | 0924 | 0013 | 0005 | 0206 | NV (0) | NZ (0) | PL (0) | 0F68 | 0126 | 13 | 05 | 31 | INT 21 |
| | 0924 | 0013 | 0005 | 0206 | NV (0) | NZ (0) | PL (0) | 0F68 | 0128 | 13 | 05 | 31 | INT 20 |

Figure 2.7: Full Program first run tracing chart

```
-e100
0F68:0100  DE.BA   E8.10   45.02   FA.B4   AC.09   AA.CD   3C.21
0D.B2
0F68:0108  75.10   FA.8A   56.1E   8B.04   36.02   92.8A   DE.0E
89.05
0F68:0110  4C.02   FE.2A   5E.D9   8E.7D   06.08   08.FE   D3.06
26.06
0F68:0118  80.02   3E.02   43.DA   04.EB   34.F6   00.88   57.1E
0F.04
0F68:0120  BA.02   42.BA   86.06   E9.02   65.B4   FE.09   BF.CD
81.21
0F68:0128  00.CD   8B.20
-e204
0F68:0204  FE.05   06.05
-e210 "Water Filling Program by Anthony Chavez" 0d 0a "$"
-e206 30 0d 0a "$"
-u100 128
0F68:0100 BA1002        MOV     DX,0210
0F68:0103 B409          MOV     AH,09
0F68:0105 CD21          INT     21
0F68:0107 B210          MOV     DL,10
0F68:0109 8A1E0402      MOV     BL,[0204]
0F68:010D 8A0E0502      MOV     CL,[0205]
0F68:0111 2AD9          SUB     BL,CL
0F68:0113 7D08          JGE     011D
0F68:0115 FE060602      INC     BYTE PTR [0206]
0F68:0119 02DA          ADD     BL,DL
0F68:011B EBF6          JMP     0113
0F68:011D 881E0402      MOV     [0204],BL
0F68:0121 BA0602        MOV     DX,0206
0F68:0124 B409          MOV     AH,09
0F68:0126 CD21          INT     21
0F68:0128 CD20          INT     20
```

Figure 2.8: Full Program second run

```
-g=100 107
Water Filling Program by Anthony Chavez

AX=0924  BX=0000  CX=0000  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0107   NV UP EI PL NZ NA PO NC
0F68:0107 B210          MOV     DL,10
-t

AX=0924  BX=0000  CX=0000  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0109   NV UP EI PL NZ NA PO NC
0F68:0109 8A1E0402      MOV     BL,[0204]
DS:0204=05
-t

AX=0924  BX=0005  CX=0000  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=010D   NV UP EI PL NZ NA PO NC
0F68:010D 8A0E0502      MOV     CL,[0205]
DS:0205=05
-t

AX=0924  BX=0005  CX=0005  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0111   NV UP EI PL NZ NA PO NC
0F68:0111 2AD9          SUB     BL,CL
-t

AX=0924  BX=0000  CX=0005  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0113   NV UP EI PL ZR NA PE NC
0F68:0113 7D08          JGE     011D
-t

AX=0924  BX=0000  CX=0005  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=011D   NV UP EI PL ZR NA PE NC
0F68:011D 881E0402      MOV     [0204],BL
DS:0204=05
-t

AX=0924  BX=0000  CX=0005  DX=0210  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0121   NV UP EI PL ZR NA PE NC
0F68:0121 BA0602        MOV     DX,0206
-t

AX=0924  BX=0000  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0124   NV UP EI PL ZR NA PE NC
0F68:0124 B409          MOV     AH,09
-t

AX=0924  BX=0000  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0126   NV UP EI PL ZR NA PE NC
0F68:0126 CD21          INT     21
```

```
-p
0

AX=0924  BX=0000  CX=0005  DX=0206  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=0F68  ES=0F68  SS=0F68  CS=0F68  IP=0128   NV UP EI PL ZR NA PE NC
0F68:0128 CD20          INT     20
-p

Program terminated normally
```

Figure 2.9: Full Program second run trace

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EEE 174-CpE 185 | | | | | | | | | | | | | |
| Laboratory Exercise #2 | | | | | | | | | Name: Anthony Chavez | | | | |
| | | | | | | | | | | | | | |
| Program Tracing Chart | | | | | | | | | | | | | |
| | | Registers: | | | | | | | | | | | |
| | | | | | | | | | | | | | |
| | AX: | BX: | CX: | DX: | OF: | ZF: | SF: | CS: | IP: | DS:204 | DS:205 | DS:206 | Next Instruction |
| Value: ---> | 0924 | 0000 | 0000 | 0210 | NV (0) | NZ (0) | PL (0) | 0F68 | 0107 | 05 | 05 | 30 | MOV DL,10 |
| | 0924 | 0000 | 0000 | 0210 | NV (0) | NZ (0) | PL (0) | 0F68 | 0109 | 05 | 05 | 30 | MOV BL,[0204] |
| | 0924 | 0005 | 0000 | 0210 | NV (0) | NZ (0) | PL (0) | 0F68 | 010D | 05 | 05 | 30 | MOV CL,[0205] |
| | 0924 | 0005 | 0005 | 0210 | NV (0) | NZ (0) | PL (0) | 0F68 | 0111 | 05 | 05 | 30 | SUB BL,CL |
| | 0924 | 0000 | 0005 | 0210 | NV (0) | ZR (1) | PL (0) | 0F68 | 0113 | 05 | 05 | 30 | JGE 011D |
| | 0924 | 0000 | 0005 | 0210 | NV (0) | ZR (1) | PL (0) | 0F68 | 011D | 05 | 05 | 30 | MOV [0204],BL |
| | 0924 | 0000 | 0005 | 0210 | NV (0) | ZR (1) | PL (0) | 0F68 | 0121 | 05 | 05 | 30 | MOV DX,0206 |
| | 0924 | 0000 | 0005 | 0206 | NV (0) | ZR (1) | PL (0) | 0F68 | 0124 | 05 | 05 | 30 | MOV AH,09 |
| | 0924 | 0000 | 0005 | 0206 | NV (0) | ZR (1) | PL (0) | 0F68 | 0126 | 05 | 05 | 30 | INT 21 |
| | 0924 | 0000 | 0005 | 0206 | NV (0) | ZR (1) | PL (0) | 0F68 | 0128 | 00 | 05 | 30 | INT 20 |

Figure 2.10: Full Program second run tracing chart

```
Full Program:
0F68:0100 BA1002        MOV     DX,0210        ; Get Program Title
0F68:0103 B409          MOV     AH,09          ; Load Program Title
0F68:0105 CD21          INT     21             ; Display Program Title
0F68:0107 B220          MOV     DL,20          ; Set Water Filling Rate
0F68:0109 8A1E0402      MOV     BL,[0204]      ; Get Current Water Level from memory
0F68:010D 8A0E0502      MOV     CL,[0205]      ; Get Irrigation Amount from memory
0F68:0111 2AD9          SUB     BL,CL          ; Subtract Irrigation Amount from
Current Water Level
0F68:0113 7D08          JGE     011D           ; Tank Filled?
0F68:0115 FE060602      INC     BYTE PTR [0206] ; Increment number of times tank is
filled by 1
0F68:0119 00DA          ADD     DL,BL          ; Fill Tank using Filling Rate
0F68:011B EBF6          JMP     0113           ; Return to Tank Filled?
0F68:011D 881E0402      MOV     [0204],BL      ; Store Current Water Level
0F68:0121 BA0602        MOV     DX,0206        ; Get Times Tank Filled Count
0F68:0124 B409          MOV     AH,09          ; Load Times Tank Filled Count
0F68:0126 CD21          INT     21             ; Display Times Tank Filled Count
0F68:0128 CD20          INT     20             ; End Program
```

Figure 2.11: Commented Full Program

```c
/*
 This programs performs the same function as the
 assembly language program used in DEBUG for Lab 1 Part 2.
 */
#include<stdio.h>

int main() {
    int mem1[3] = {20, 50, 0};        // Memory Addresses 0204, 0205, 0206
    char mem2[1][40] = {"Water Filling Program by Anthony Chavez"};  // Memory Addresses 0210 ...

    printf("%s\n", mem2[0]);          // Display Program Title and Name

    int DL = 20,          // Register DL ; MOV DL, ##  ; Set Water Filling Rate
        BL = mem1[0],     // Register BL ; MOV BL,[##] ; Get Current Water Level from memory
        CL = mem1[1];     // Register CL ; MOV CL,[##] ; Get Irrigation Amount from memory

    printf("Memory 0204 is: %d\n", mem1[0]);
    printf("Memory 0205 is: %d\n", mem1[1]);
    printf("Memory 0206 is: %d\n", mem1[2]);
    printf("Value of DL register: %d\n", DL);
    printf("Value in BL register: %d\n", BL);
    printf("Value in CL register: %d\n", CL);

    BL = BL - CL;       // SUB BL,CL ; Subtract Irrigation Amount from Current Water Level

    printf("Value in BL register: %d\n", BL);

    while(BL < 0) {                  // JGE 011D ; Tank Filled?
        mem1[2] = mem1[2] + 1;       // INC BYTE PTR [0206] ; Increment number of times tank is filled by 1
        printf("Count incremented\n");
        BL = BL + DL;                // ADD BL,DL ; Fill Tank using Filling Rate
        printf("Value in BL register: %d\n", BL);
    }

    mem1[0] = BL;                     // MOV [0204],BL ; Store Current Water Level
    printf("Value in Memory 0200: %d\n", mem1[0]);

    printf("Counter: %d\n", mem1[2]);  // Display Times Tank Filled Count

    return 0;                          // INT 20 ; End Program
}
```

```
----jGRASP exec: C:\Users\[redacted]
Water Filling Program by Anthony Chavez
Memory 0204 is: 20
Memory 0205 is: 50
Memory 0206 is: 0
Value of DL register: 20
Value in BL register: 20
Value in CL register: 50
Value in BL register: -30
Count incremented
Value in BL register: -10
Count incremented
Value in BL register: 10
Value in Memory 0200: 10
Counter: 2

----jGRASP: operation complete.
```

Figure 2.12: Converted Full Program from Assembly to C

## Part 3: Microsoft's Assembly Language Development System (MASM)

For this part of the lab, I was introduced to the MASM assembler and gained experience in the syntax, pragmatics of the Programmer's Workbench (PWB), and the Code View (Debugger). First, the original program provided has a runtime error when using the DOS DEBUG (see Figure 3.2). When running the debugger, the contents in memory are not automatically reset in between runs so I added a reset procedure to do so (see Figure 3.4, 3.5, 3.7,3.9,3.10). Second, the program was modified to all the user to specify the number of times the program loops (see Figure 3.12, 3.14, 3.15). Third, I converted the program to pure C which only requires a few lines of code to achieve the same output (see Figure 3.16 and 3.17). Last, I converted the program to C and Inline Assembly which also achieves the same output result (see Figure 3.18).

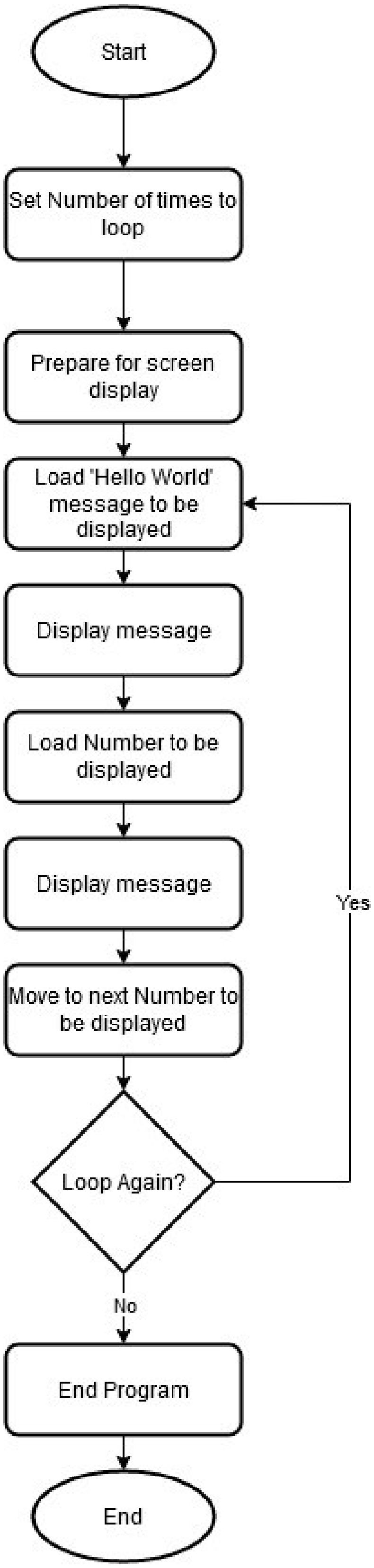


Figure 3.1: Original Program with Runtime Error Flow Chart

```
;***********************
;*                     *
;* MASM Hello          *
;*                     *
;***********************
cseg segment 'code'
assume cs:cseg, ds:cseg, ss:cseg, es:cseg

        org 100h                    ; organize instructions to start at mem loc 100

start:  mov cx,10                   ; Number of loops
        mov ah,9                    ; Prepare for screen display

again:  mov dx, offset Hello        ; Load 'Hello World' message to be displayed
        int 21h                                     ; Display message to screen
        mov dx, offset Num_msg      ; Load Number to be displayed
        int 21h                     ; Display Number to screen
        inc byte ptr Num_msg        ; Move to next Number to be displayed
        loopne again                ; If Num of loops isn't zero keep looping

done:   mov ah, 4ch                 ; End Program
        int 21h

        org 200h                    ; organize instructions to start at mem location 200

Hello   db "Hello World", 20h, 20h, "$"

Num_msg db 30h, 13, 10, 36

cseg ends
end start
```

Figure 3.2: Original Program with Runtime Error Commented Code

```
Microsoft (R) Macro Assembler Version 6.14.8444                07/13/20 16:46:53
HELLO1.ASM                                                      Page 1 - 1

                                ;***********************
                                ;*                    *
                                ;* MASM Hello         *
                                ;*                    *
                                ;***********************
 0000                           cseg segment 'code'
                                assume cs:cseg, ds:cseg, ss:cseg, es:cseg

                                        org 100h

 0100  B9 000A                  start:  mov cx,10
 0103  B4 09                            mov ah,9

 0105  BA 0200 R                again:  mov dx, offset Hello
 0108  CD 21                            int 21h
 010A  BA 020E R                        mov dx, offset Num_msg
 010D  CD 21                            int 21h
 010F  FE 06 020E R                     inc byte ptr Num_msg
 0113  E0 F0                            loopne again

 0115  B4 4C                    done:   mov ah, 4ch
 0117  CD 21                            int 21h

                                        org 200h

 0200 48 65 6C 6C 6F 20         Hello   db "Hello World", 20h, 20h, "$"
      57 6F 72 6C 64 20
      20 24



 020E 30 0D 0A 24               Num_msg db 30h, 13, 10, 36

 0212                           cseg ends
                                end start

Microsoft (R) Macro Assembler Version 6.14.8444                07/13/20 16:46:53
HELLO1.ASM                                                      Symbols 2 - 1
```

```
Segments and Groups:

            N a m e              Size    Length   Align   Combine Class

cseg . . . . . . . . . . . . .   16 Bit   0212     Para     Private 'CODE'


Symbols:

            N a m e              Type    Value    Attr

Hello  . . . . . . . . . . . .   Byte     0200     cseg
Num_msg  . . . . . . . . . . .            Byte     020E     cseg
again  . . . . . . . . . . . .   L Near   0105     cseg
done . . . . . . . . . . . . .   L Near   0115     cseg
start  . . . . . . . . . . . .   L Near   0100     cseg

             0 Warnings
             0 Errors
```

Figure 3.3: Original Program with Runtime Error List File

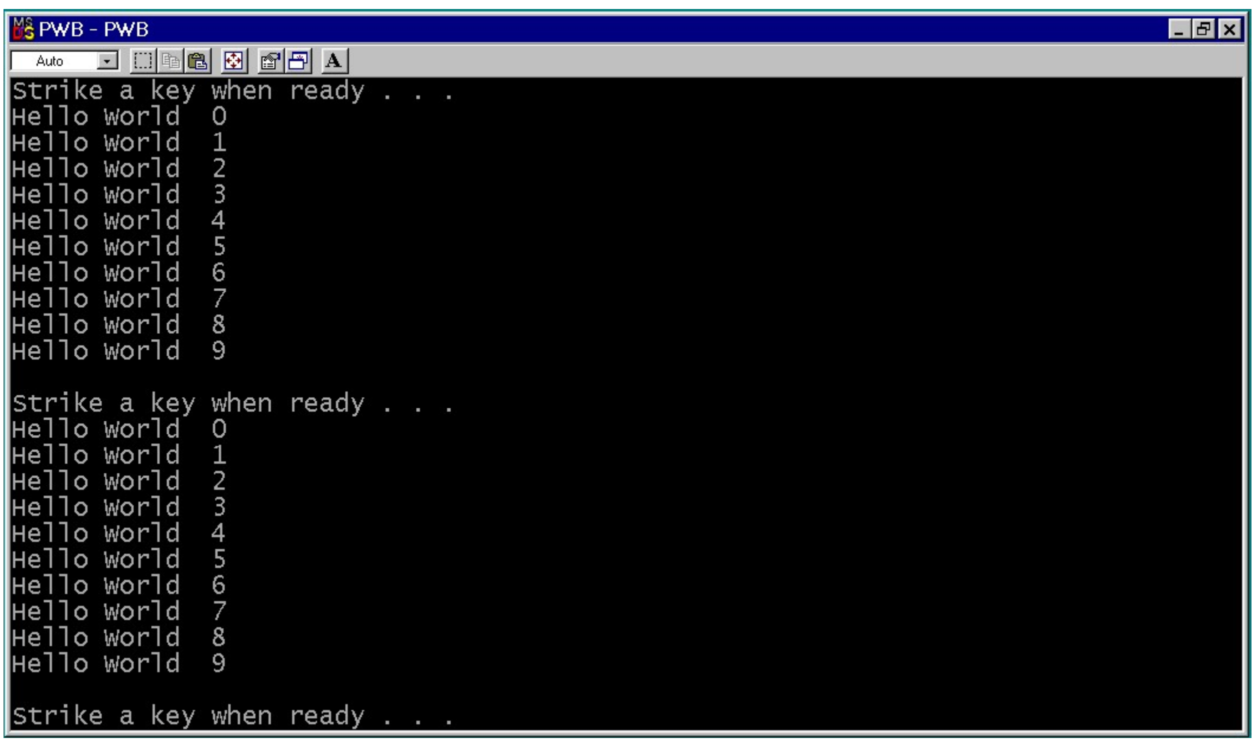



Figure 3.4: Original Program with Runtime Error Output MASM Run 1 and 2

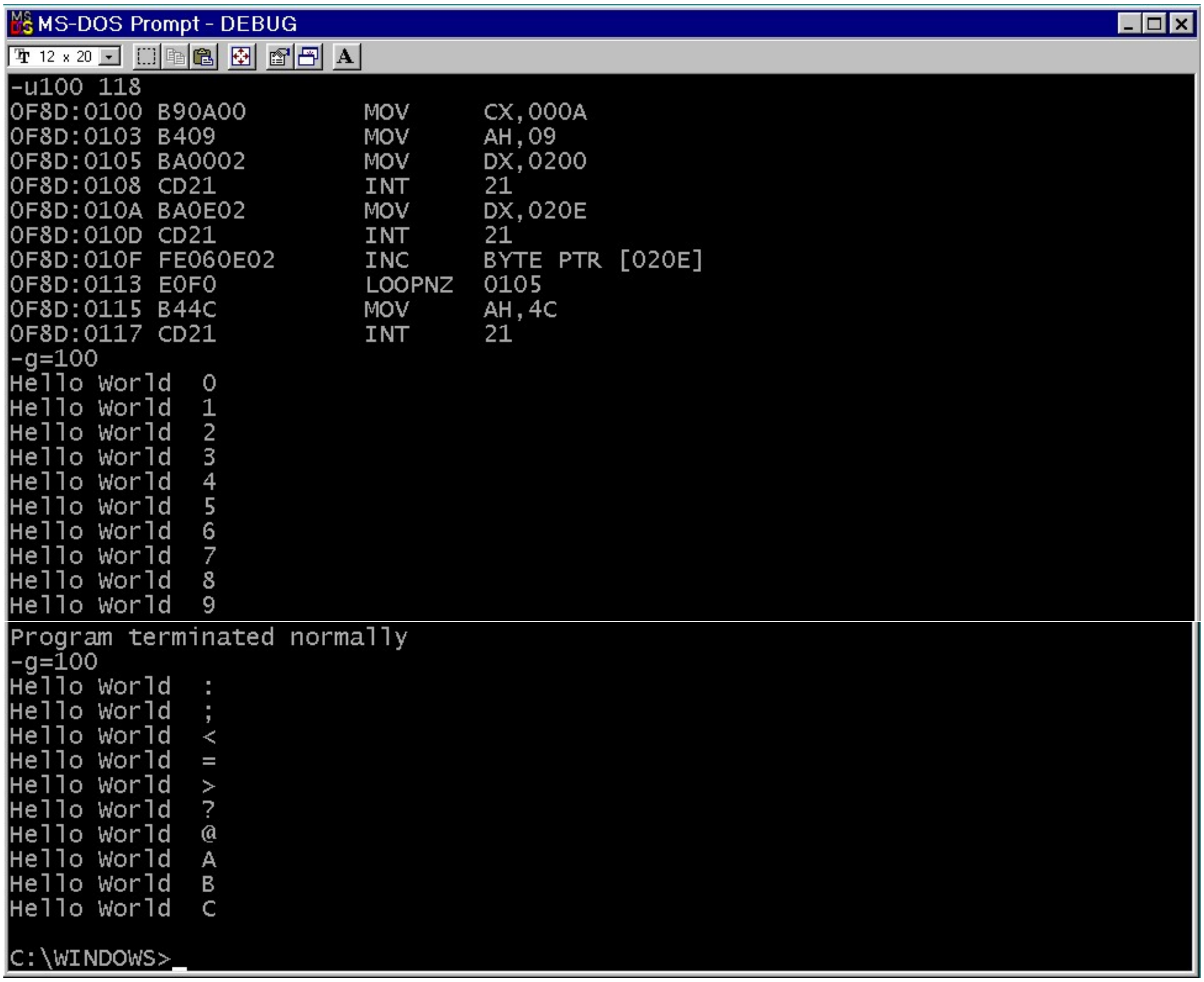


Figure 3.5: Original Program with Runtime Error Output DOS DEBUG, Run 1 and 2

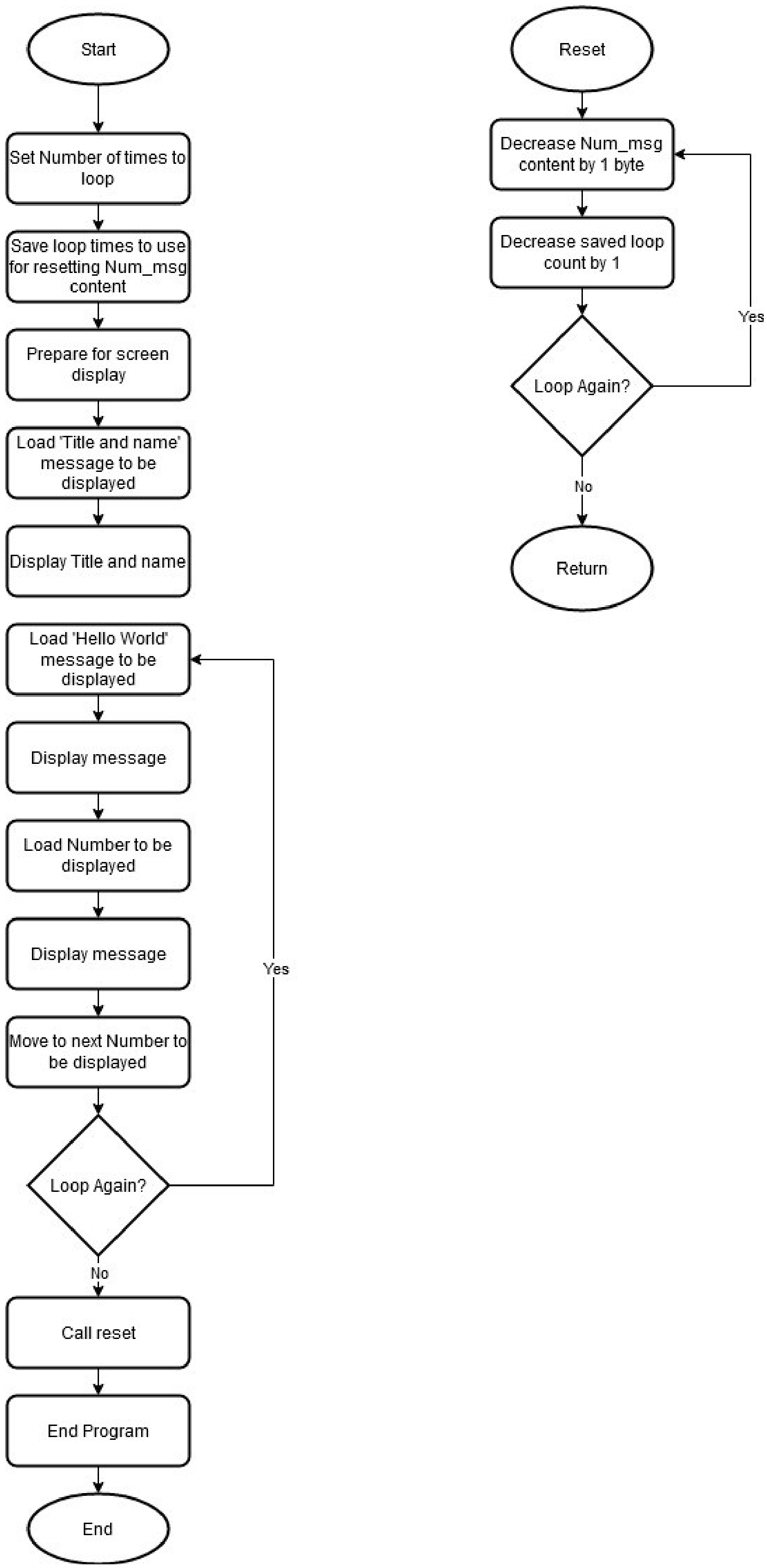



Figure 3.6: Modified Program 2 Flow Chart

```
;***********************
;*                    *
;* MASM Hello         *
;*                    *
;***********************
cseg segment 'code'
assume cs:cseg, ds:cseg, ss:cseg, es:cseg

        org 100h                    ; organize instructions to start at mem location 100

start:  mov cx,10                   ; Number of loops
        mov bx,cx                   ; Store number of loops for resetting procedure
        mov ah,9                    ; Prepare for screen display
        mov dx, offset msg          ; Load 'MASM Hello and name' message to be displayed
        int 21h                     ; Display Program title and my name

again:  mov dx, offset Hello        ; Load 'Hello World' message to be displayed
        int 21h                     ; Display message to screen
        mov dx, offset Num_msg      ; Load Number to be displayed
        int 21h                     ; Display Number to screen
        inc byte ptr Num_msg        ; Move to next Number to be displayed
        loopne again                ; If Num of loops isn't zero keep looping

done:   call reset                  ; Reset the Num_msg to 30h
        mov ah, 4ch                 ; End Program
        int 21h

reset:  dec byte ptr Num_msg        ; Decrement Num_msg by 1 byte
        dec bx                      ; Decrement saved number of loops by 1
        cmp bx,0                    ; Check if saved number of loops is 0
        jne reset                   ; If saved number of loops isn’t zero keep looping
        ret                         ; Return back to ‘done’ procedure

        org 200h                    ; organize instructions to start at mem location 200

msg     db "MASM Hello by Anthony Chavez", 0ah,0dh, "$"

Hello   db "Hello World", 20h, 20h, "$"

Num_msg db 30h, 13, 10, 36

cseg ends
end start
```

Figure 3.7: Modified Program 2 Commented Code

```
Microsoft (R) Macro Assembler Version 6.14.8444                 07/14/20 01:12:37
HELLO1_2.ASM                                                    Page 1 - 1


                                 ;***********************
                                 ;*                 *
                                 ;* MASM Hello         *
                                 ;*                 *
                                 ;***********************
 0000                            cseg segment 'code'
                                 assume cs:cseg, ds:cseg, ss:cseg, es:cseg

                                         org 100h

 0100  B9 000A                   start:  mov cx,10
 0103  8B D9                             mov bx,cx
 0105  B4 09                             mov ah,9
 0107  BA 0200 R                         mov dx, offset msg
 010A  CD 21                             int 21h

 010C  BA 021F R                 again:  mov dx, offset Hello
 010F  CD 21                             int 21h
 0111  BA 022D R                         mov dx, offset Num_msg
 0114  CD 21                             int 21h
 0116  FE 06 022D R                      inc byte ptr Num_msg
 011A  E0 F0                             loopne again

 011C  E8 0004                   done:   call reset
 011F  B4 4C                             mov ah, 4ch
 0121  CD 21                             int 21h

 0123  FE 0E 022D R              reset:  dec byte ptr Num_msg
 0127  4B                                dec bx
 0128  83 FB 00                          cmp bx,0
 012B  75 F6                             jne reset
 012D  C3                                ret

                                         org 200h


 0200 4D 41 53 4D 20 48                  msg     db "MASM Hello by Anthony Chavez", 0ah,0dh, "$"
      65 6C 6C 6F 20 62
      79 20 41 6E 74 68
      6F 6E 79 20 43 68
      61 76 65 7A 0A 0D
      24
```

```
021F 48 65 6C 6C 6F 20          Hello    db "Hello World", 20h, 20h, "$"
     57 6F 72 6C 64 20
     20 24

 022D 30 0D 0A 24               Num_msg db 30h, 13, 10, 36

 0231                           cseg ends
                                end start

Microsoft (R) Macro Assembler Version 6.14.8444                    07/14/20 01:12:37
HELLO1_2.ASM                                                Symbols 2 - 1




Segments and Groups:

            N a m e              Size     Length   Align   Combine Class

cseg . . . . . . . . . . . . . .    16 Bit    0231     Para     Private 'CODE'


Symbols:

            N a m e              Type     Value    Attr

Hello  . . . . . . . . . . . . .    Byte      021F     cseg
Num_msg  . . . . . . . . . . . .          Byte      022D     cseg
again  . . . . . . . . . . . . .    L Near    010C     cseg
done . . . . . . . . . . . . . .    L Near    011C     cseg
msg  . . . . . . . . . . . . . .    Byte      0200     cseg
reset  . . . . . . . . . . . . .    L Near    0123     cseg
start  . . . . . . . . . . . . .    L Near    0100     cseg

           0 Warnings
           0 Errors
```
Figure 3.8: Modified Program 2 List File

```
Strike a key when ready . . .
MASM Hello by Anthony Chavez
Hello World  0
Hello World  1
Hello World  2
Hello World  3
Hello World  4
Hello World  5
Hello World  6
Hello World  7
Hello World  8
Hello World  9

Strike a key when ready . . . _
```

Figure 3.9: Modified Program 2 Output MASM

```
-g=100
MASM Hello by Anthony Chavez
Hello World  0
Hello World  1
Hello World  2
Hello World  3
Hello World  4
Hello World  5
Hello World  6
Hello World  7
Hello World  8
Hello World  9
```

```
-g=100
MASM Hello by Anthony Chavez
Hello World  0
Hello World  1
Hello World  2
Hello World  3
Hello World  4
Hello World  5
Hello World  6
Hello World  7
Hello World  8
Hello World  9
```

Figure 3.10: Modified Program 2 Output DOS DEBUG, Run 1 and 2

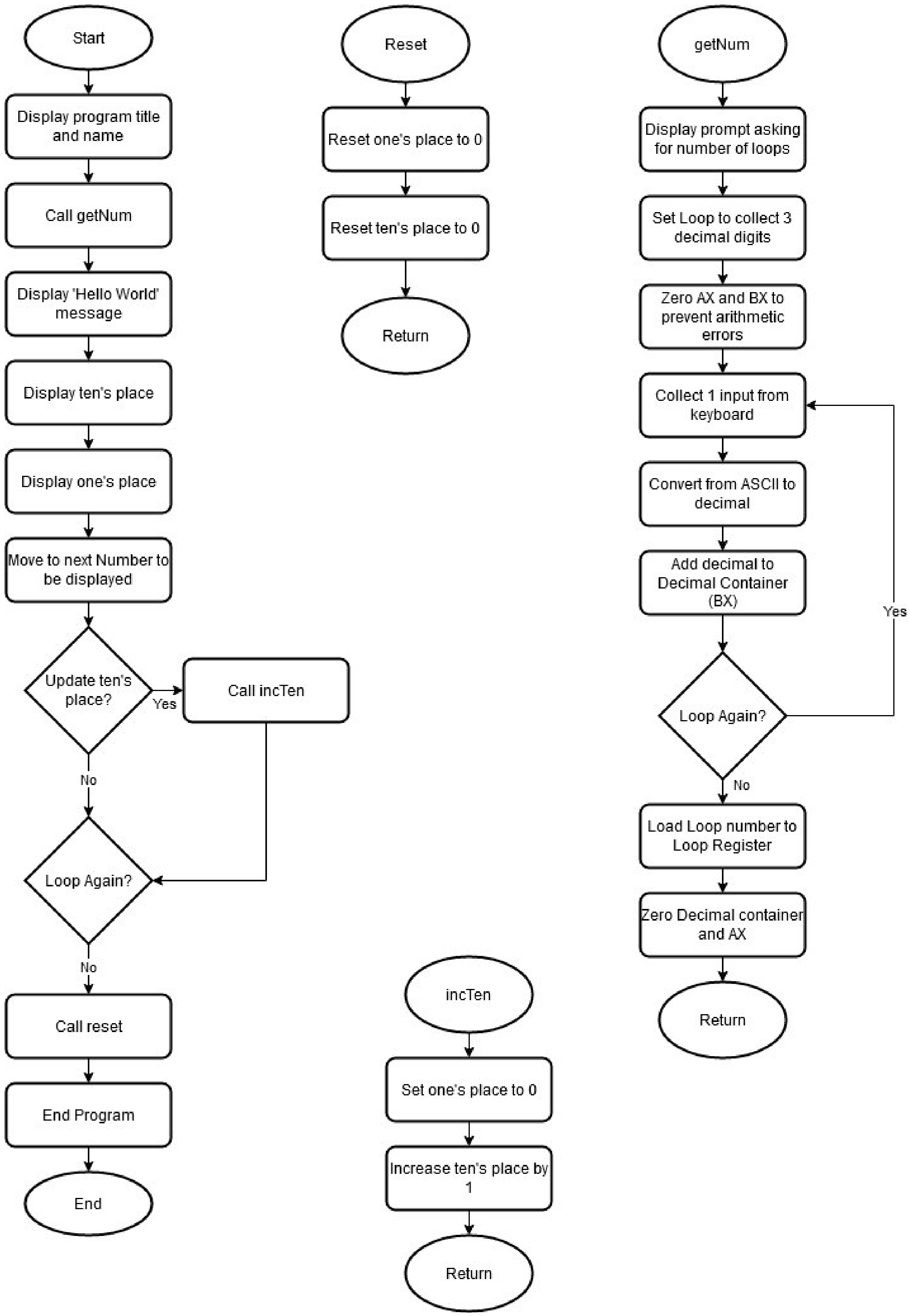



Figure 3.11: Modified Program 3 Flowchart

```
;***********************
;*                     *
;* MASM Hello          *
;*                     *
;***********************
cseg segment 'code'
assume cs:cseg, ds:cseg, ss:cseg, es:cseg

        org 100h                    ; organize instructions to start at mem location 100

start:  mov ah,9                    ; Prepare for screen display
        mov dx, offset msg          ; Load 'MASM Hello and name' message to be displayed
        int 21h                     ; Display Program title and my name
        call getNum                 ; Get User Input for loop count



again:  mov dx, offset Hello        ; Load 'Hello World' message to be displayed
        int 21h                     ; Display message to screen
        mov dx, offset Tens_msg     ; Load ten's place
        int 21h                     ; Display ten's place
        mov dx, offset Num_msg      ; Load Number to be displayed
        int 21h                     ; Display Number to screen
        inc byte ptr Num_msg        ; Move to next Number to be displayed
        mov bl, byte ptr Num_msg
        cmp bl,3Ah                  ; Check if tens place needs to be updated
        je incTen
back:   loopne again                ; If Num of loops isn't zero keep looping

done:   call reset                  ; Reset Num_msg and Tens_msg to 0
        mov ah, 4ch                 ; End Program
        int 21h

incTen: mov dl,30h
        mov byte ptr Num_msg,dl     ; Reset one's place to 0
        inc byte ptr Tens_msg       ; Increase ten's place by 1
        jmp back

getNum:  mov dx, offset prmpt       ; Ask user to input how many times the loop runs
         int 21h                    ; Display prmpt
         mov cx,3                   ; Set Loop to collect 3 decimal digits
         mov ax,0                   ; Zero AX to prevent arithmetic errors
         mov bx,0                   ; Container to hold decimal digits
```

```
top:    mov ax,10
        mul bx
        mov bx,ax
        mov ah,1                    ; Function to read a char from keyboard
        int 21h                     ; The char saved in AL
        sub al,30h                  ; Convert digit from ASCII to decimal
        mov ah,0
        add bx,ax                   ; Add digit to create decimal number
        loop top
        mov cx,bx                   ; Load number of loops
        mov bx,0                    ; Zero Container to hold decimal digits
        mov ax,0                    ; Zero AX
        mov ah,9
        mov dx, offset nextLine     ; Bring cursor to a new line
        int 21h
        ret                         ; Return to 'start' procedure

reset:  mov dx,0
        mov dl,30h
        mov byte ptr Num_msg,dl     ; Reset one's place to 0
        mov byte ptr Tens_msg,dl    ; Reset ten's place to 0
        ret                         ; Return to 'done' procedure

        org 200h                    ; Organize instructions to start at mem location 200

msg     db "MASM Hello by Anthony Chavez", 0ah,0dh, "$"

prmpt   db "How many times do you wish to loop", 0ah,0dh, "$"

nextLine db 0ah,0dh, "$"

Hello   db "Hello World", 20h, 20h, "$"

Tens_msg db 30h, 36

Num_msg db 30h, 13, 10, 36

cseg ends
end start
```

Figure 3.12: Modified Program 3 Commented Code

```
                            ;************************
                            ;*                *
                            ;* MASM Hello         *
                            ;*                *
                            ;************************
 0000                       cseg segment 'code'
                            assume cs:cseg, ds:cseg, ss:cseg, es:cseg

                                    org 100h             ; organize instructions to start at mem
location 100

 0100  B4 09                start:  mov ah,9             ; Prepare for screen display
 0102  BA 0200 R            mov dx, offset msg       ; Load 'MASM Hello and name' message to be
displayed
 0105  CD 21                        int 21h                      ; Display Program title and my
name
 0107  E8 0031                      call getNum                  ; Get User Input for loop count



 010A  BA 0247 R       again:  mov dx, offset Hello         ; Load 'Hello World' message to be displayed
 010D  CD 21                        int 21h                      ; Display message to screen
 010F  BA 0255 R            mov dx, offset Tens_msg      ; Load ''
 0112  CD 21                        int 21h
 0114  BA 0257 R            mov dx, offset Num_msg       ; Load Number to be displayed
 0117  CD 21                        int 21h                      ; Display Number to screen
 0119  FE 06 0257 R         inc byte ptr Num_msg         ; Move to next Number to be displayed
 011D  8A 1E 0257 R                 mov bl, byte ptr Num_msg
 0121  80 FB 3A                     cmp bl,3Ah                   ; Check if tens place needs to be
updated
 0124  74 09                        je incTen
 0126  E0 E2                back:   loopne again                 ; If Num of loops isn't zero keep
looping

 0128  E8 0041              done:   call reset           ; Reset Num_msg and Tens_msg to 0
 012B  B4 4C                        mov ah, 4ch                  ; End Program
 012D  CD 21                        int 21h

 012F  B2 30                incTen: mov dl,30h
 0131  88 16 0257 R         mov byte ptr Num_msg,dl  ; Reset one's place to 0
 0135  FE 06 0255 R         inc byte ptr Tens_msg        ; Increase ten's place by 1
 0139  EB EB                        jmp back

 013B  BA 021F R       getNum: mov dx, offset prmpt         ; Ask user to input how many times the loop
runs
 013E  CD 21                        int 21h                      ; Display prmpt
 0140  B9 0003                      mov cx,3             ; Set Loop to collect 3 decimal digits
 0143  B8 0000                      mov ax,0             ; Zero AX to prevent arithmetic errors
 0146  BB 0000                      mov bx,0             ; Container to hold decimal digits
 0149  B8 000A              top:    mov ax,10
 014C  F7 E3                        mul bx
 014E  8B D8                        mov bx,ax
 0150  B4 01                        mov ah,1             ; Function to read a char from keyboard
```

```
0152  CD 21                     int 21h                 ; The char saved in AL
 0154  2C 30                    sub al,30h              ; Convert digit from ASCII to
decimal
 0156  B4 00                    mov ah,0
 0158  03 D8                    add bx,ax               ; Add digit to create decimal
number
 015A  E2 ED                    loop top
 015C  8B CB                    mov cx,bx
 015E  BB 0000                  mov bx,0
 0161  B8 0000                  mov ax,0
 0164  B4 09                    mov ah,9
 0166  BA 0244 R           mov dx, offset nextLine      ; Bring cursor to a new line
 0169  CD 21                    int 21h
 016B  C3                  ret                          ; Return to 'start' procedure

 016C  BA 0000             reset:  mov dx,0
 016F  B2 30                    mov dl,30h
 0171  88 16 0257 R        mov byte ptr Num_msg,dl      ; Reset one's place to 0
 0175  88 16 0255 R        mov byte ptr Tens_msg,dl     ; Reset ten's place to 0
 0179  C3                  ret

                                org 200h                ; Organize instructions to start at mem
location 200

 0200 4D 41 53 4D 20 48    msg      db "MASM Hello by Anthony Chavez", 0ah,0dh, "$"
      65 6C 6C 6F 20 62
      79 20 41 6E 74 68
      6F 6E 79 20 43 68
      61 76 65 7A 0A 0D
      24

 021F 48 6F 77 20 6D 61    prmpt    db "How many times do you wish to loop", 0ah,0dh, "$"
      6E 79 20 74 69 6D
      65 73 20 64 6F 20
      79 6F 75 20 77 69
      73 68 20 74 6F 20
      6C 6F 6F 70 0A 0D
      24

 0244 0A 0D 24             nextLine db 0ah,0dh, "$"

 0247 48 65 6C 6C 6F 20    Hello    db "Hello World", 20h, 20h, "$"
      57 6F 72 6C 64 20
      20 24

 0255 30 24                Tens_msg db 30h, 36

 0257 30 0D 0A 24       Num_msg db 30h, 13, 10, 36

 025B                      cseg ends
                           end start
```

```
Microsoft (R) Macro Assembler Version 6.14.8444          07/14/20 11:58:03
HELLO1_3.ASM                                              Symbols 2 - 1



Segments and Groups:

          N a m e          Size    Length   Align   Combine Class

cseg . . . . . . . . . . . . . .    16 Bit    025B    Para    Private 'CODE'


Symbols:

          N a m e          Type    Value    Attr

Hello  . . . . . . . . . . . . .    Byte      0247    cseg
Num_msg  . . . . . . . . . . . .    Byte      0257    cseg
Tens_msg . . . . . . . . . . . .    Byte      0255    cseg
again  . . . . . . . . . . . . .    L Near    010A    cseg
back . . . . . . . . . . . . . .    L Near    0126    cseg
done . . . . . . . . . . . . . .    L Near    0128    cseg
getNum . . . . . . . . . . . . .    L Near    013B    cseg
incTen . . . . . . . . . . . . .    L Near    012F    cseg
msg  . . . . . . . . . . . . . .    Byte      0200    cseg
nextLine . . . . . . . . . . . .    Byte      0244    cseg
prmpt  . . . . . . . . . . . . .    Byte      021F    cseg
reset  . . . . . . . . . . . . .    L Near    016C    cseg
start  . . . . . . . . . . . . .    L Near    0100    cseg
top  . . . . . . . . . . . . . .    L Near    0149    cseg

           0 Warnings
           0 Errors
```

Figure 3.13: Modified Program 3 List File

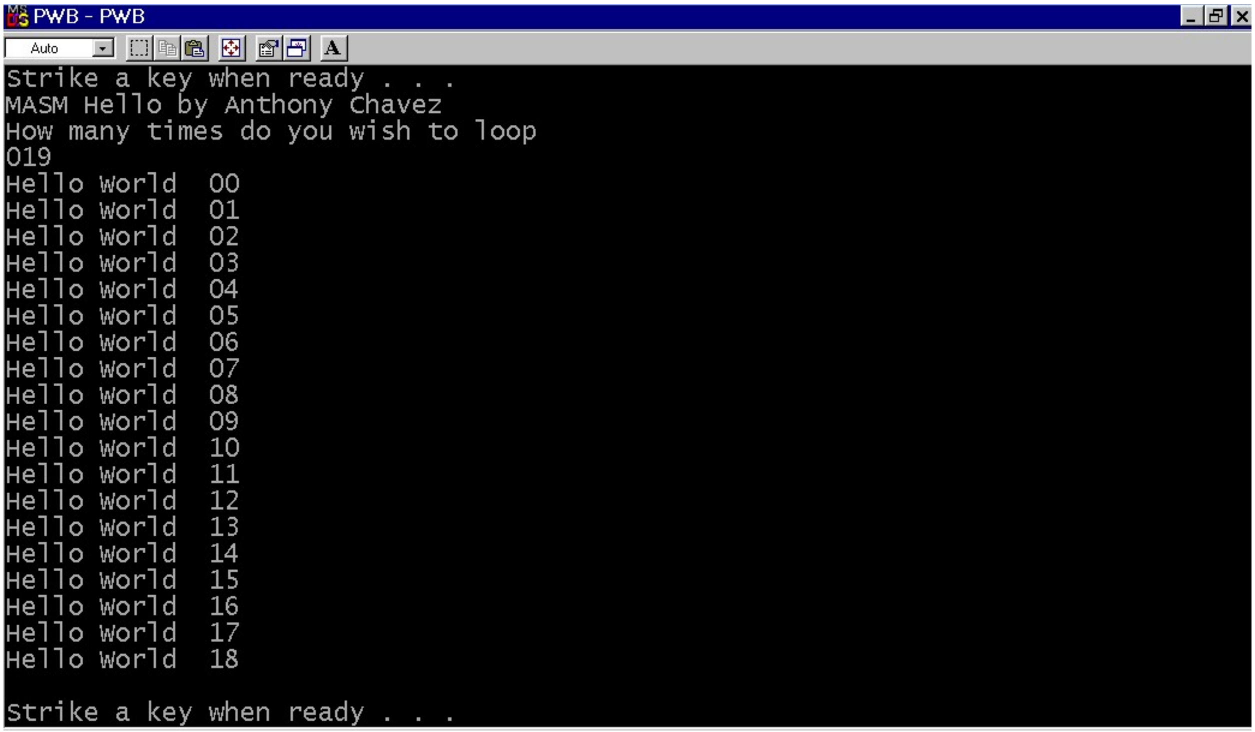
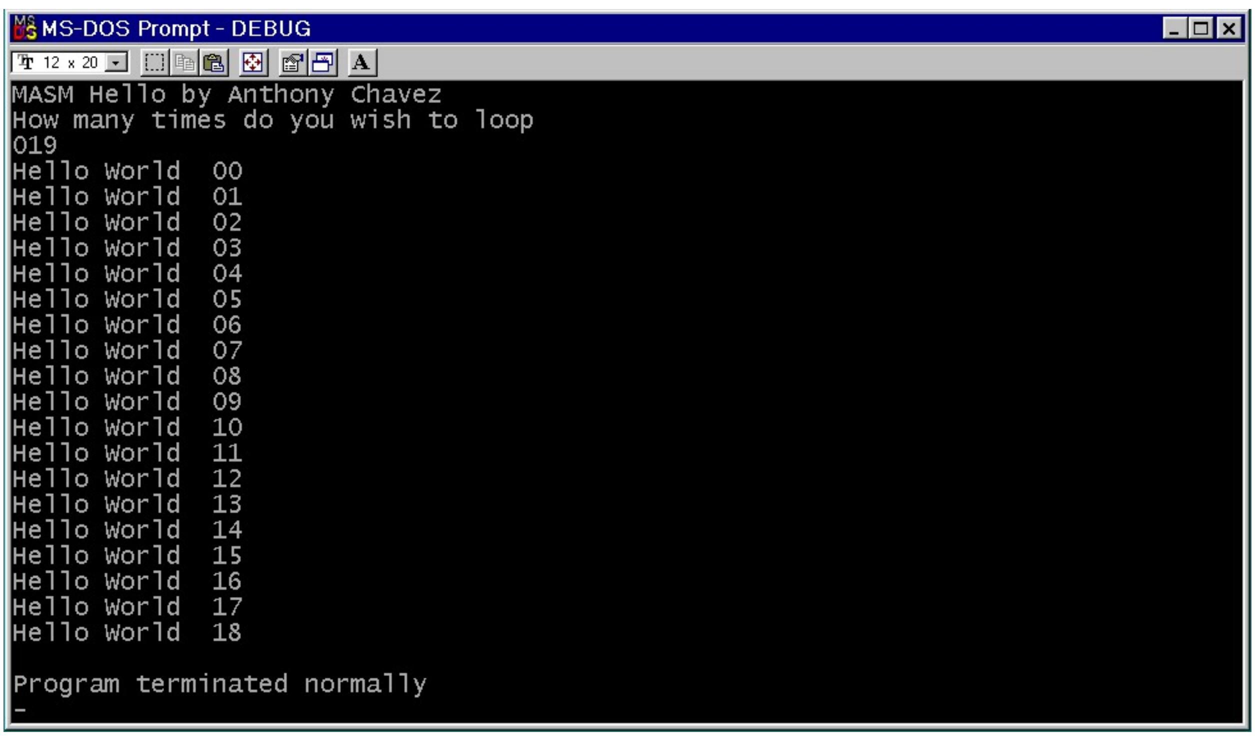


Figure 3.14: Modified Program 3 Output MASM



```
Program terminated normally
-g=100
MASM Hello by Anthony Chavez
How many times do you wish to loop
015
Hello World  00
Hello World  01
Hello World  02
Hello World  03
Hello World  04
Hello World  05
Hello World  06
Hello World  07
Hello World  08
Hello World  09
Hello World  10
Hello World  11
Hello World  12
Hello World  13
Hello World  14
```

Figure 3.15: Modified Program 3 Output DOS DEBUG, Run 1 and 2

```c
/*
 This programs performs the same function as the
 assembly language program used in MASM for Part
 3 of lab 1.
 */
#include<stdio.h>

int main() {
    printf("MASM Hello by Anthony Chavez\n");          // Display Program Title and Name

    int loopNum = 0;    // Number of loops
    int num = 1;        // Number msg

    printf("How many times do you wish to loop: ");
    scanf("%d", &loopNum);      // Get number of loops desired

    while(loopNum > 0) {
        printf("Hello World  ");
        printf("%d\n", num);
        loopNum = loopNum - 1;  // Decrement loop number
        num = num + 1;          // Increment Number msg
    }

    return 0;
}
```

Figure 3.16: Modified Program 3 Converted to Pure C

```
MASM Hello by Anthony Chavez
How many times do you wish to loop: 99
Hello World  1
Hello World  2
Hello World  3
Hello World  4
Hello World  5
Hello World  6
Hello World  7
Hello World  8
Hello World  9
Hello World  10
Hello World  11
Hello World  12
Hello World  13
Hello World  14
Hello World  15
Hello World  16
Hello World  17
Hello World  18
Hello World  19
Hello World  20
Hello World  21
Hello World  22
Hello World  23
Hello World  24
Hello World  25
Hello World  26
Hello World  27
Hello World  28
Hello World  29
Hello World  30
Hello World  31
Hello World  32
Hello World  33
Hello World  34
Hello World  35
Hello World  36
Hello World  37
Hello World  38
Hello World  39
Hello World  40
Hello World  41
Hello World  42
Hello World  43
Hello World  44
Hello World  45
Hello World  46
Hello World  47
Hello World  48
Hello World  49
Hello World  50
Hello World  51
Hello World  52
Hello World  53
Hello World  54
Hello World  55
Hello World  56
Hello World  57
Hello World  58
Hello World  59
Hello World  60
Hello World  61
Hello World  62
Hello World  63
Hello World  64
Hello World  65
Hello World  66
Hello World  67
Hello World  68
Hello World  69
Hello World  70
Hello World  71
Hello World  72
Hello World  73
Hello World  74
Hello World  75
Hello World  76
Hello World  77
Hello World  78
Hello World  79
Hello World  80
Hello World  81
Hello World  82
Hello World  83
Hello World  84
Hello World  85
Hello World  86
Hello World  87
Hello World  88
Hello World  89
Hello World  90
Hello World  91
Hello World  92
Hello World  93
Hello World  94
Hello World  95
Hello World  96
Hello World  97
Hello World  98
Hello World  99
```

Figure 3.17: Modified Program 3 Converted to Pure C Output

```c
/*
 This programs performs the same function as the
 assembly language program used in MASM for Part
 3 of lab 1.
 */
#include<stdio.h>

int increment(int a);
int decrement(int a);

int main() {
    printf("MASM Hello by Anthony Chavez\n");          // Display Program Title and Name

    int loopNum = 0;    // Number of loops
    int num = 1;        // Number msg

    printf("How many times do you wish to loop: ");
    scanf("%d", &loopNum);      // Get number of loops desired

    while(loopNum > 0) {
        printf("Hello World  ");
        printf("%d\n", num);
        loopNum = decrement(loopNum);    // Decrement loop number
        num = increment(num);    // Increment Number msg
    }

    return 0;
}

int increment(int a) {
    int c;

    __asm(
           "mov %1, %%eax;"
           "inc %%eax;"
           "mov %%eax, %0;"
           :"=r"(c)
           :"r"(a)
           :"%eax"
           );
           return c;
}

int decrement(int a) {
  int c;

  __asm(
        "mov %1, %%eax;"
        "dec %%eax;"
        "mov %%eax, %0;"
        :"=r"(c)
        :"r"(a)
        :"%eax"
        );
        return c;
}
```

Figure 3.18: Modified Program 3 Converted to C and Inline Assembly

Conclusions:

First, the “dump” command (“d”) displays the contents of the memory locations. There are 3 data columns: the left side shows the Code Segment (CS) and the Instruction Pointer (IP), the middle is the data stored in memory, and the right side is the data stored in the middle trying to be converted to a printable character in ASCII. The addresses are in ascending 4 digit hexadecimal order but displayed every 3 decimals and the data segments are 2 digit hexadecimal and there are 16 in each row with the half way point marked by a “-“. Second, the “enter” command (“e”) is used to enter the assembly language program and change the data in the memory. Third, the “unassembled” command (“u”) is used to see the program you just entered, and it will be converted to assembly language. Fourth, the “register modify” command (“r”) is used to set the Instruction Pointer (IP) register to point to a given address location. Fifth, the “trace” command (“t”) is used to step through a program you just entered where you can see all the register values and the next instruction to be executed. Finally, the “go” command (“g”) is used to run the code until it reaches a specified address (breakpoint).

As for the assembly language mnemonics, the “move” command (MOV) is used to enter data from another register or address, the “subtraction” command (SUB) performs subtraction with the syntax being Dest. = Dest. – Source, the “addition” command (ADD) performs similar to SUB but adds the numbers, the “jump greater than or equal” command (JGE) will jump to the specified address if the AX register is greater than or equal to zero (positive), and the “interrupt 20” command gives control to the processor to end the program.

Overall, the lab took a little longer than I initially expected even with my previous experience with using MASM from another class. I feel a lot more comfortable with using all the tools I learned in this lab except for Inline Assembly. I hope I get to avoid working with this confusing language in the future.