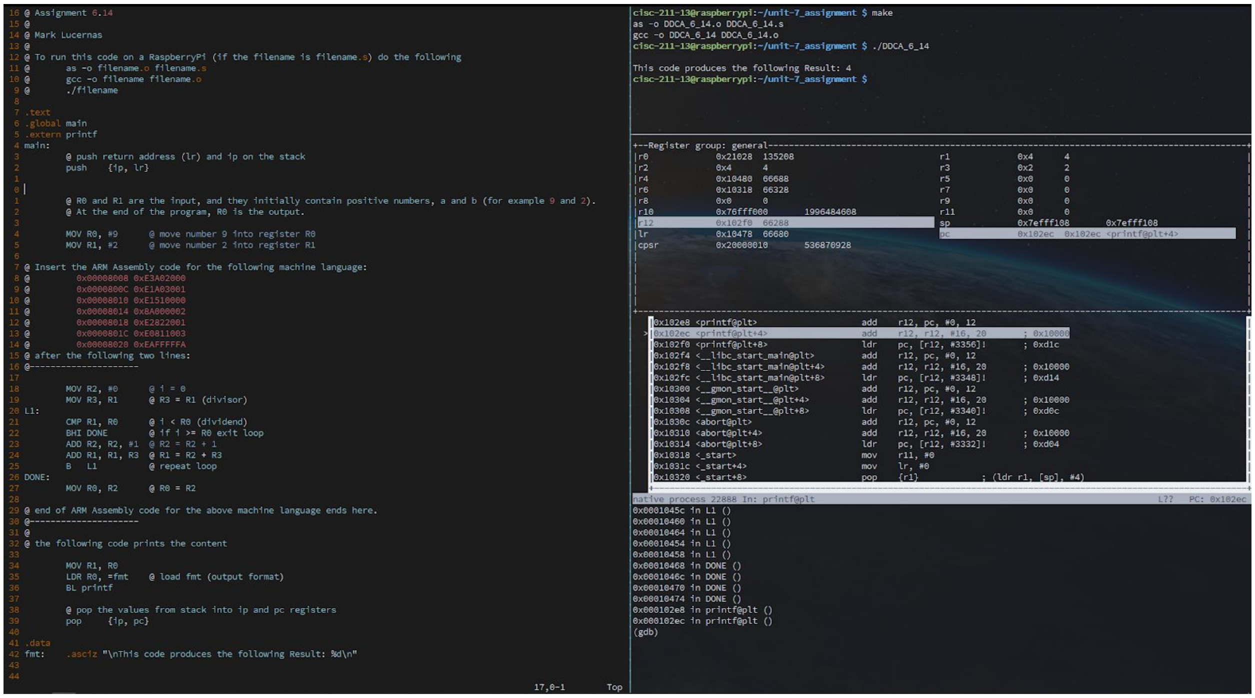
Mark Lucernas
CISC 211
Prof. Saied Moezzi
December 5, 2020

cisc-211-13@raspberrypi

Unit 7 Assignment

Overview Screen Capture

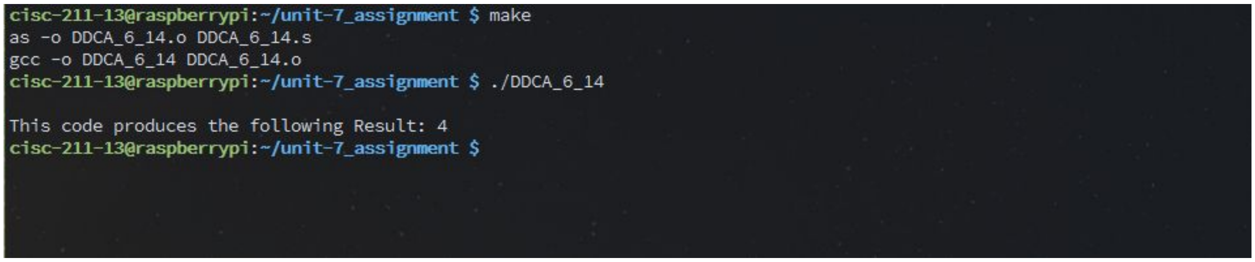


```
@ Assignment 6.14
@
@ Mark Lucernas
@
@ To run this code on a RaspberryPi (if the filename is filename.s) do the following
@       as -o filename.o filename.s
@       gcc -o filename filename.o
@       ./filename

.text
.global main
.extern printf
main:
        @ push return address (lr) and ip on the stack
        push    {ip, lr}


        @ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).
        @ At the end of the program, R0 is the output.

        MOV R0, #9      @ move number 9 into register R0
        MOV R1, #2      @ move number 2 into register R1

@ Insert the ARM Assembly code for the following machine language:
@         0x00008008 0xE3A02000
@         0x0000800C 0xE1A03001
@         0x00008010 0xE1510000
@         0x00008014 0x8A000002
@         0x00008018 0xE2822001
@         0x0000801C 0xE0811003
@         0x00008020 0xEAFFFFFA
@ after the following two lines:
@--------------------

        MOV R2, #0      @ i = 0
        MOV R3, R1      @ R3 = R1 (divisor)
L1:
        CMP R1, R0      @ i < R0 (dividend)
        BHI DONE        @ if i >= R0 exit loop
        ADD R2, R2, #1  @ R2 = R2 + 1
        ADD R1, R1, R3  @ R1 = R2 + R3
        B   L1          @ repeat loop
DONE:
        MOV R0, R2      @ R0 = R2

@ end of ARM Assembly code for the above machine language ends here.
@--------------------
@
@ the following code prints the content

        MOV R1, R0
        LDR R0, =fmt    @ load fmt (output format)
        BL printf

        @ pop the values from stack into ip and pc registers
        pop     {ip, pc}

.data
fmt:    .asciz "\nThis code produces the following Result: %d\n"


                                                          17,0-1        Top
```

```
cisc-211-13@raspberrypi:~/unit-7_assignment $ make
as -o DDCA_6_14.o DDCA_6_14.s
gcc -o DDCA_6_14 DDCA_6_14.o
cisc-211-13@raspberrypi:~/unit-7_assignment $ ./DDCA_6_14

This code produces the following Result: 4
cisc-211-13@raspberrypi:~/unit-7_assignment $
```

```
+--Register group: general-----------------------------------------------------------------------+
|r0             0x21028  135208                          r1             0x4      4                |
|r2             0x4      4                               r3             0x2      2                |
|r4             0x10480  66688                           r5             0x0      0                |
|r6             0x10318  66328                           r7             0x0      0                |
|r8             0x0      0                               r9             0x0      0                |
|r10            0x76fff000       1996484608              r11            0x0      0                |
|r12            0x102f0  66288                           sp             0x7efff108       0x7efff108 |
|lr             0x10478  66680                           pc             0x102ec  0x102ec <printf@plt+4> |
|cpsr           0x20000010       536870928                                                         |
+------------------------------------------------------------------------------------------------+
   |0x102e8 <printf@plt>                    add    r12, pc, #0, 12
  >|0x102ec <printf@plt+4>                  add    r12, r12, #16, 20      ; 0x10000
   |0x102f0 <printf@plt+8>                  ldr    pc, [r12, #3356]!      ; 0xd1c
   |0x102f4 <__libc_start_main@plt>         add    r12, pc, #0, 12
   |0x102f8 <__libc_start_main@plt+4>       add    r12, r12, #16, 20      ; 0x10000
   |0x102fc <__libc_start_main@plt+8>       ldr    pc, [r12, #3348]!      ; 0xd14
   |0x10300 <__gmon_start__@plt>            add    r12, pc, #0, 12
   |0x10304 <__gmon_start__@plt+4>          add    r12, r12, #16, 20      ; 0x10000
   |0x10308 <__gmon_start__@plt+8>          ldr    pc, [r12, #3340]!      ; 0xd0c
   |0x1030c <abort@plt>                     add    r12, pc, #0, 12
   |0x10310 <abort@plt+4>                   add    r12, r12, #16, 20      ; 0x10000
   |0x10314 <abort@plt+8>                   ldr    pc, [r12, #3332]!      ; 0xd04
   |0x10318 <_start>                        mov    r11, #0
   |0x1031c <_start+4>                      mov    lr, #0
   |0x10320 <_start+8>                      pop    {r1}           ; (ldr r1, [sp], #4)
native process 22888 In: printf@plt                                          L??   PC: 0x102ec
0x0001045c in L1 ()
0x00010460 in L1 ()
0x00010464 in L1 ()
0x00010454 in L1 ()
0x00010458 in L1 ()
0x00010468 in DONE ()
0x0001046c in DONE ()
0x00010470 in DONE ()
0x00010474 in DONE ()
0x000102e8 in printf@plt ()
0x000102ec in printf@plt ()
(gdb)
```

Compile & Program Execution Screen Capture

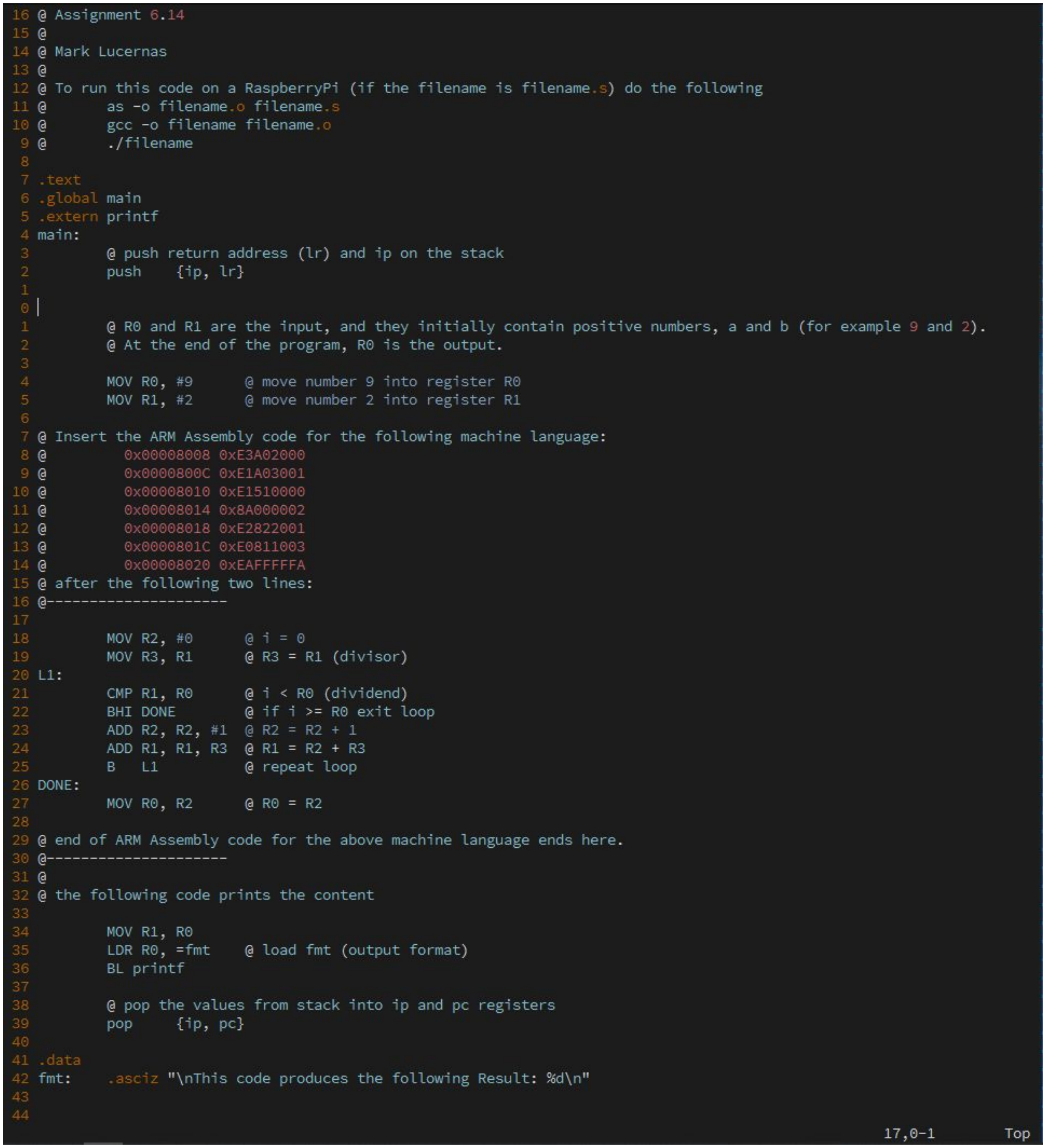


```
cisc-211-13@raspberrypi:~/unit-7_assignment $ make
as -o DDCA_6_14.o DDCA_6_14.s
gcc -o DDCA_6_14 DDCA_6_14.o
cisc-211-13@raspberrypi:~/unit-7_assignment $ ./DDCA_6_14

This code produces the following Result: 4
cisc-211-13@raspberrypi:~/unit-7_assignment $
```

Description:

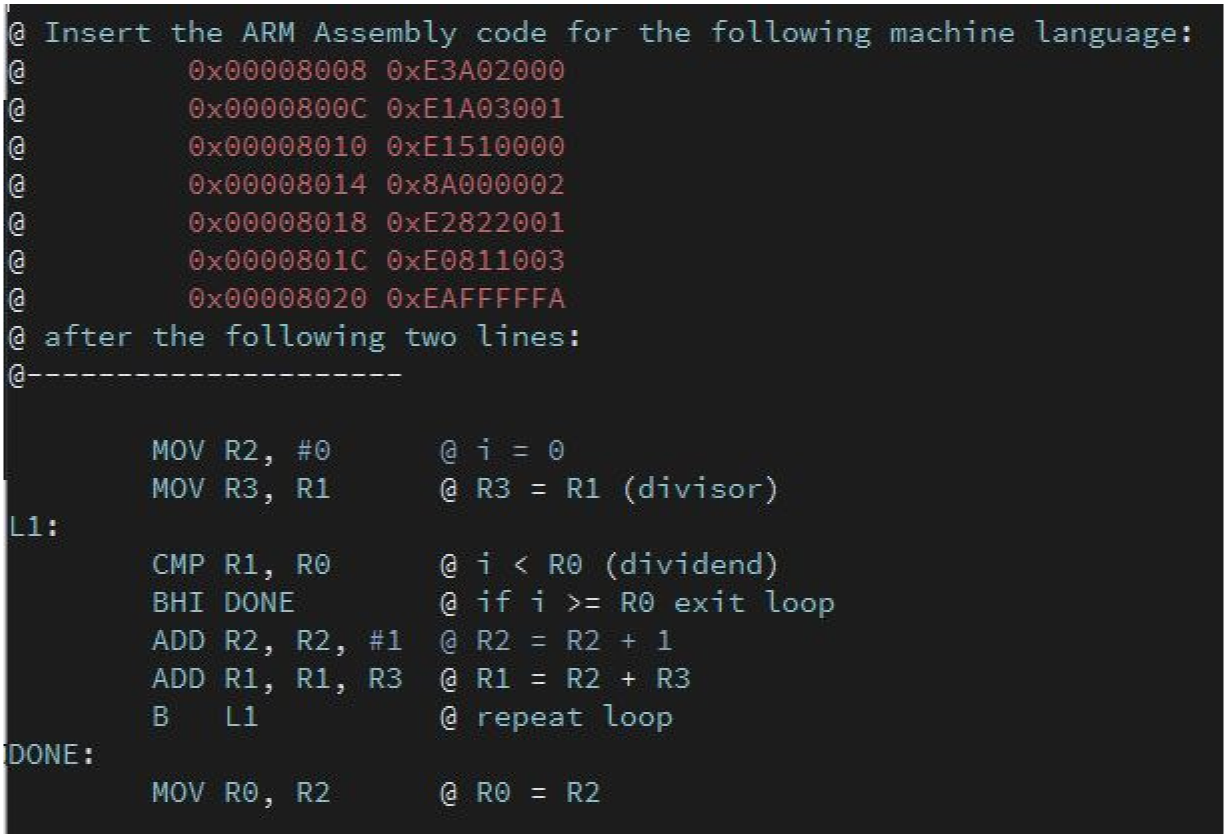
This program executes integer division (A / B = Quotient Output).

Assembly Source Code Screen Capture



```
@ Assignment 6.14
@
@ Mark Lucernas
@
@ To run this code on a RaspberryPi (if the filename is filename.s) do the following
@       as -o filename.o filename.s
@       gcc -o filename filename.o
@       ./filename

.text
.global main
.extern printf
main:
        @ push return address (lr) and ip on the stack
        push    {ip, lr}


        @ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).
        @ At the end of the program, R0 is the output.

        MOV R0, #9      @ move number 9 into register R0
        MOV R1, #2      @ move number 2 into register R1

@ Insert the ARM Assembly code for the following machine language:
@         0x00008008 0xE3A02000
@         0x0000800C 0xE1A03001
@         0x00008010 0xE1510000
@         0x00008014 0x8A000002
@         0x00008018 0xE2822001
@         0x0000801C 0xE0811003
@         0x00008020 0xEAFFFFFA
@ after the following two lines:
@---------------------

        MOV R2, #0      @ i = 0
        MOV R3, R1      @ R3 = R1 (divisor)
L1:
        CMP R1, R0      @ i < R0 (dividend)
        BHI DONE        @ if i >= R0 exit loop
        ADD R2, R2, #1  @ R2 = R2 + 1
        ADD R1, R1, R3  @ R1 = R2 + R3
        B   L1          @ repeat loop
DONE:
        MOV R0, R2      @ R0 = R2

@ end of ARM Assembly code for the above machine language ends here.
@---------------------
@
@ the following code prints the content

        MOV R1, R0
        LDR R0, =fmt    @ load fmt (output format)
        BL printf

        @ pop the values from stack into ip and pc registers
        pop     {ip, pc}

.data
fmt:    .asciz "\nThis code produces the following Result: %d\n"


                                                                                   17,0-1        Top
```

Edited Assembly Code Screen Capture



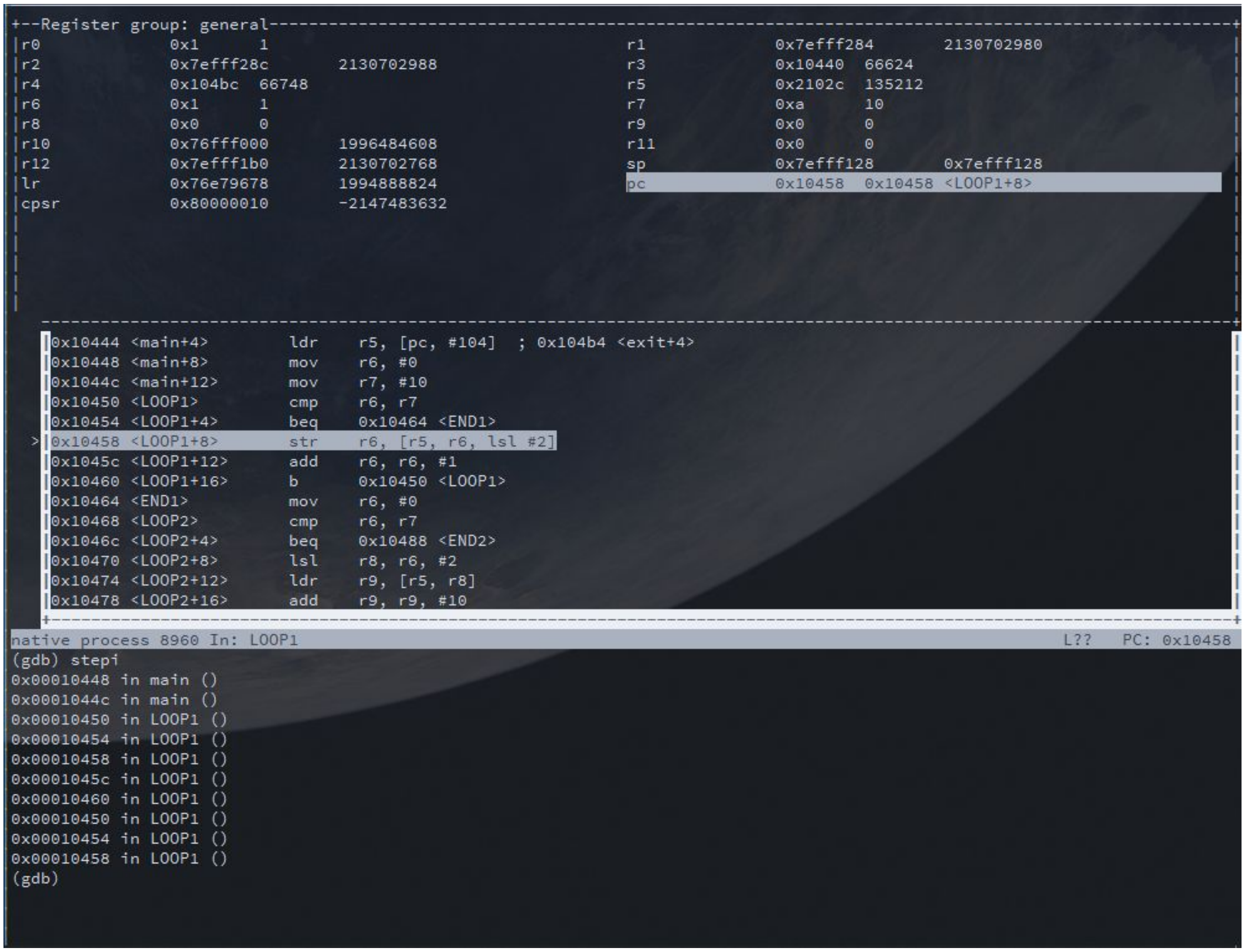
```
@ Insert the ARM Assembly code for the following machine language:
@         0x00008008 0xE3A02000
@         0x0000800C 0xE1A03001
@         0x00008010 0xE1510000
@         0x00008014 0x8A000002
@         0x00008018 0xE2822001
@         0x0000801C 0xE0811003
@         0x00008020 0xEAFFFFFA
@ after the following two lines:
@---------------------

        MOV R2, #0      @ i = 0
        MOV R3, R1      @ R3 = R1 (divisor)
L1:
        CMP R1, R0      @ i < R0 (dividend)
        BHI DONE        @ if i >= R0 exit loop
        ADD R2, R2, #1  @ R2 = R2 + 1
        ADD R1, R1, R3  @ R1 = R2 + R3
        B   L1          @ repeat loop
DONE:
        MOV R0, R2      @ R0 = R2
```

DDCA_unit-7_assignment.s

```
@ Assignment 6.14
@
@ Mark Lucernas
@
@ To run this code on a RaspberryPi (if the filename is filename.s) do the following
@       as -o filename.o filename.s
@       gcc -o filename filename.o
@       ./filename

.text
.global main
.extern printf
main:
        @ push return address (lr) and ip on the stack
        push    {ip, lr}


        @ R0 and R1 are the input, and they initially contain positive numbers, a and b (for example 9 and 2).
        @ At the end of the program, R0 is the output.

        MOV R0, #9   @ move number 9 into register R0
        MOV R1, #2   @ move number 2 into register R1

@ Insert the ARM Assembly code for the following machine language:
@       0x00008008 0xE3A02000
@       0x0000800C 0xE1A03001
@       0x00008010 0xE1510000
@       0x00008014 0x8A000002
@       0x00008018 0xE2822001
@       0x0000801C 0xE0811003
@       0x00008020 0xEAFFFFFA
@ after the following two lines:
@---------------------

        MOV R2, #0   @ i = 0
        MOV R3, R1   @ R3 = R1 (divisor)
L1:
        CMP R1, R0   @ i < R0 (dividend)
        BHI DONE     @ if i >= R0 exit loop
        ADD R2, R2, #1  @ R2 = R2 + 1
        ADD R1, R1, R3  @ R1 = R2 + R3
        B   L1       @ repeat loop
DONE:
        MOV R0, R2   @ R0 = R2

@ end of ARM Assembly code for the above machine language ends here.
@---------------------
@
```

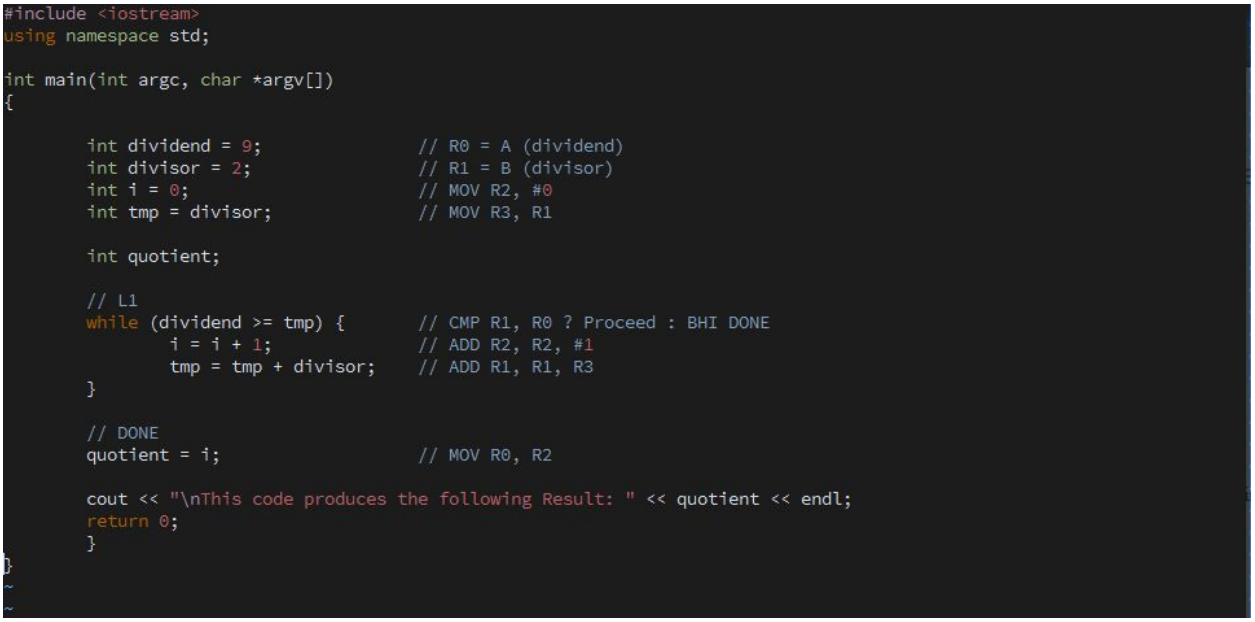
```
@ the following code prints the content

        MOV R1, R0
        LDR R0, =fmt @ load fmt (output format)
        BL printf

        @ pop the values from stack into ip and pc registers
        pop     {ip, pc}

.data
fmt:    .asciz "\nThis code produces the following Result: %d\n"
```

GDB Debugger Screen Capture



```
+--Register group: general------------------------------------------------------------------------------+
|r0             0x1      1                                     r1             0x7efff284       2130702980 |
|r2             0x7efff28c       2130702988                    r3             0x10440  66624              |
|r4             0x104bc  66748                                 r5             0x2102c  135212             |
|r6             0x1      1                                     r7             0xa      10                 |
|r8             0x0      0                                     r9             0x0      0                  |
|r10            0x76fff000       1996484608                    r11            0x0      0                  |
|r12            0x7efff1b0       2130702768                    sp             0x7efff128       0x7efff128 |
|lr             0x76e79678       1994888824                    pc             0x10458  0x10458 <LOOP1+8>  |
|cpsr           0x80000010       -2147483632                                                            |
|                                                                                                       |
|                                                                                                       |
|                                                                                                       |
|                                                                                                       |
|                                                                                                       |
   -----------------------------------------------------------------------------------------------------+
   |0x10444 <main+4>        ldr    r5, [pc, #104]   ; 0x104b4 <exit+4>                                 |
   |0x10448 <main+8>        mov    r6, #0                                                              |
   |0x1044c <main+12>       mov    r7, #10                                                             |
   |0x10450 <LOOP1>         cmp    r6, r7                                                              |
   |0x10454 <LOOP1+4>       beq    0x10464 <END1>                                                      |
  >|0x10458 <LOOP1+8>       str    r6, [r5, r6, lsl #2]                                                |
   |0x1045c <LOOP1+12>      add    r6, r6, #1                                                          |
   |0x10460 <LOOP1+16>      b      0x10450 <LOOP1>                                                     |
   |0x10464 <END1>          mov    r6, #0                                                              |
   |0x10468 <LOOP2>         cmp    r6, r7                                                              |
   |0x1046c <LOOP2+4>       beq    0x10488 <END2>                                                      |
   |0x10470 <LOOP2+8>       lsl    r8, r6, #2                                                          |
   |0x10474 <LOOP2+12>      ldr    r9, [r5, r8]                                                        |
   |0x10478 <LOOP2+16>      add    r9, r9, #10                                                         |
   +----------------------------------------------------------------------------------------------------+
native process 8960 In: LOOP1                                                            L??   PC: 0x10458
(gdb) stepi
0x00010448 in main ()
0x0001044c in main ()
0x00010450 in LOOP1 ()
0x00010454 in LOOP1 ()
0x00010458 in LOOP1 ()
0x0001045c in LOOP1 ()
0x00010460 in LOOP1 ()
0x00010450 in LOOP1 ()
0x00010454 in LOOP1 ()
0x00010458 in LOOP1 ()
(gdb)
```

C++ Source Code Screen Capture



```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{

        int dividend = 9;               // R0 = A (dividend)
        int divisor = 2;                // R1 = B (divisor)
        int i = 0;                      // MOV R2, #0
        int tmp = divisor;              // MOV R3, R1

        int quotient;

        // L1
        while (dividend >= tmp) {       // CMP R1, R0 ? Proceed : BHI DONE
                i = i + 1;              // ADD R2, R2, #1
                tmp = tmp + divisor;    // ADD R1, R1, R3
        }

        // DONE
        quotient = i;                   // MOV R0, R2

        cout << "\nThis code produces the following Result: " << quotient << endl;
        return 0;
        }
}
~
~
```

DDCA_unit-7.cpp

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{

        int dividend = 9;                       // R0 = A (dividend)
        int divisor = 2;                // R1 = B (divisor)
        int i = 0;                      // MOV R2, #0
        int tmp = divisor;                      // MOV R3, R1

        int quotient;

        // L1
        while (dividend >= tmp) {       // CMP R1, R0 ? Proceed : BHI DONE
                i = i + 1;              // ADD R2, R2, #1
                tmp = tmp + divisor;    // ADD R1, R1, R3
        }
        // DONE
        quotient = i;            // MOV R0, R2

        cout << "\nThis code produces the following Result: " << quotient << endl;
        return 0;
        }
}
```