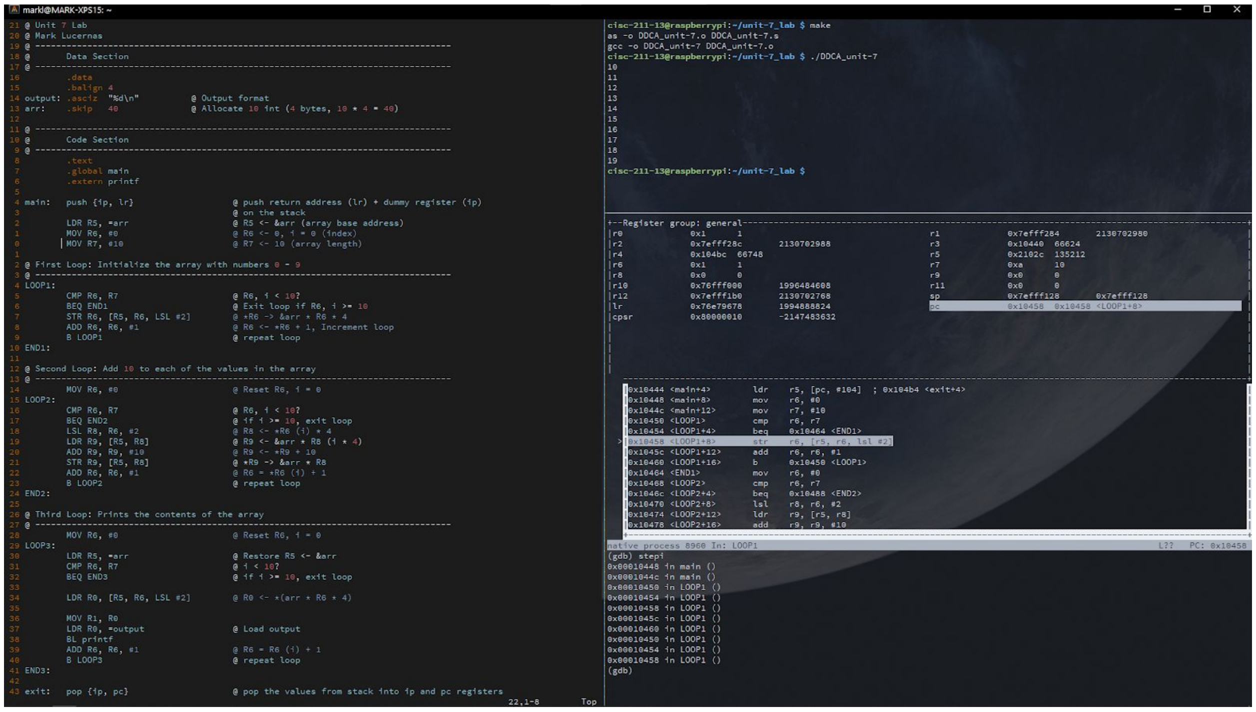
Mark Lucernas
CISC 211
Prof. Saied Moezzi
December 5, 2020

cisc-211-13@raspberrypi

Unit 7 Lab

Overview Screen Capture

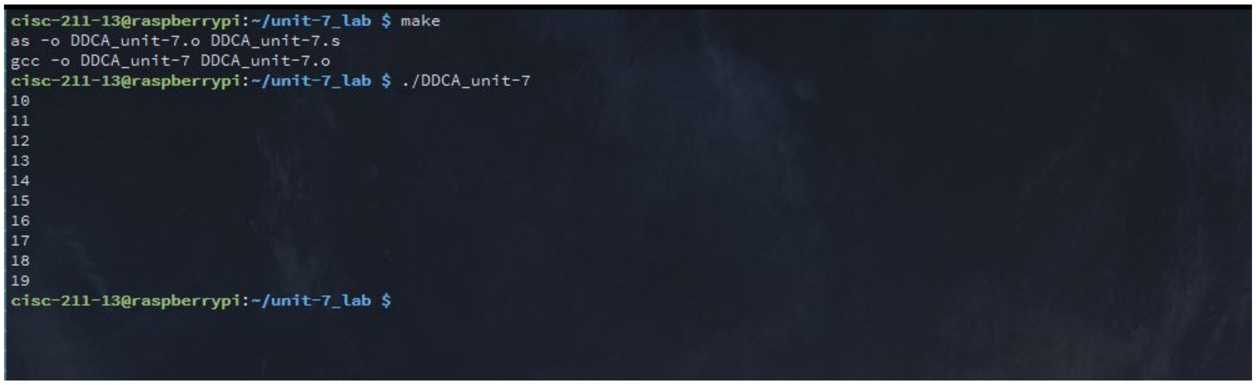


```
markl@MARK-XPS15: ~
@ Unit 7 Lab
@ Mark Lucernas
@ ---------------------------------------------------------------------------------
@       Data Section
@ ---------------------------------------------------------------------------------
        .data
        .balign 4
output: .asciz  "%d\n"          @ Output format
arr:    .skip   40              @ Allocate 10 int (4 bytes, 10 * 4 = 40)

@ ---------------------------------------------------------------------------------
@       Code Section
@ ---------------------------------------------------------------------------------
        .text
        .global main
        .extern printf

main:   push {ip, lr}                   @ push return address (lr) + dummy register (ip)
                                        @ on the stack
        LDR R5, =arr                    @ R5 <- &arr (array base address)
        MOV R6, #0                      @ R6 <- 0, i = 0 (index)
        MOV R7, #10                     @ R7 <- 10 (array length)

@ First Loop: Initialize the array with numbers 0 - 9
@ ---------------------------------------------------------------------------------
LOOP1:
        CMP R6, R7                      @ R6, i < 10?
        BEQ END1                        @ Exit loop if R6, i >= 10
        STR R6, [R5, R6, LSL #2]        @ *R6 -> &arr * R6 * 4
        ADD R6, R6, #1                  @ R6 <- *R6 + 1, Increment loop
        B LOOP1                         @ repeat loop
END1:

@ Second Loop: Add 10 to each of the values in the array
@ ---------------------------------------------------------------------------------
        MOV R6, #0                      @ Reset R6, i = 0
LOOP2:
        CMP R6, R7                      @ R6, i < 10?
        BEQ END2                        @ if i >= 10, exit loop
        LSL R8, R6, #2                  @ R8 <- *R6 (i) * 4
        LDR R9, [R5, R8]                @ R9 <- &arr * R8 (i * 4)
        ADD R9, R9, #10                 @ R9 <- *R9 + 10
        STR R9, [R5, R8]                @ *R9 -> &arr * R8
        ADD R6, R6, #1                  @ R6 = *R6 (i) + 1
        B LOOP2                         @ repeat loop
END2:

@ Third Loop: Prints the contents of the array
@ ---------------------------------------------------------------------------------
        MOV R6, #0                      @ Reset R6, i = 0
LOOP3:
        LDR R5, =arr                    @ Restore R5 <- &arr
        CMP R6, R7                      @ i < 10?
        BEQ END3                        @ if i >= 10, exit loop

        LDR R0, [R5, R6, LSL #2]        @ R0 <- *(arr * R6 * 4)

        MOV R1, R0
        LDR R0, =output                 @ Load output
        BL printf
        ADD R6, R6, #1                  @ R6 = R6 (i) + 1
        B LOOP3                         @ repeat loop
END3:

exit:   pop {ip, pc}                    @ pop the values from stack into ip and pc registers
                                                                        22,1-8        Top
```

```
cisc-211-13@raspberrypi:~/unit-7_lab $ make
as -o DDCA_unit-7.o DDCA_unit-7.s
gcc -o DDCA_unit-7 DDCA_unit-7.o
cisc-211-13@raspberrypi:~/unit-7_lab $ ./DDCA_unit-7
10
11
12
13
14
15
16
17
18
19
cisc-211-13@raspberrypi:~/unit-7_lab $
```

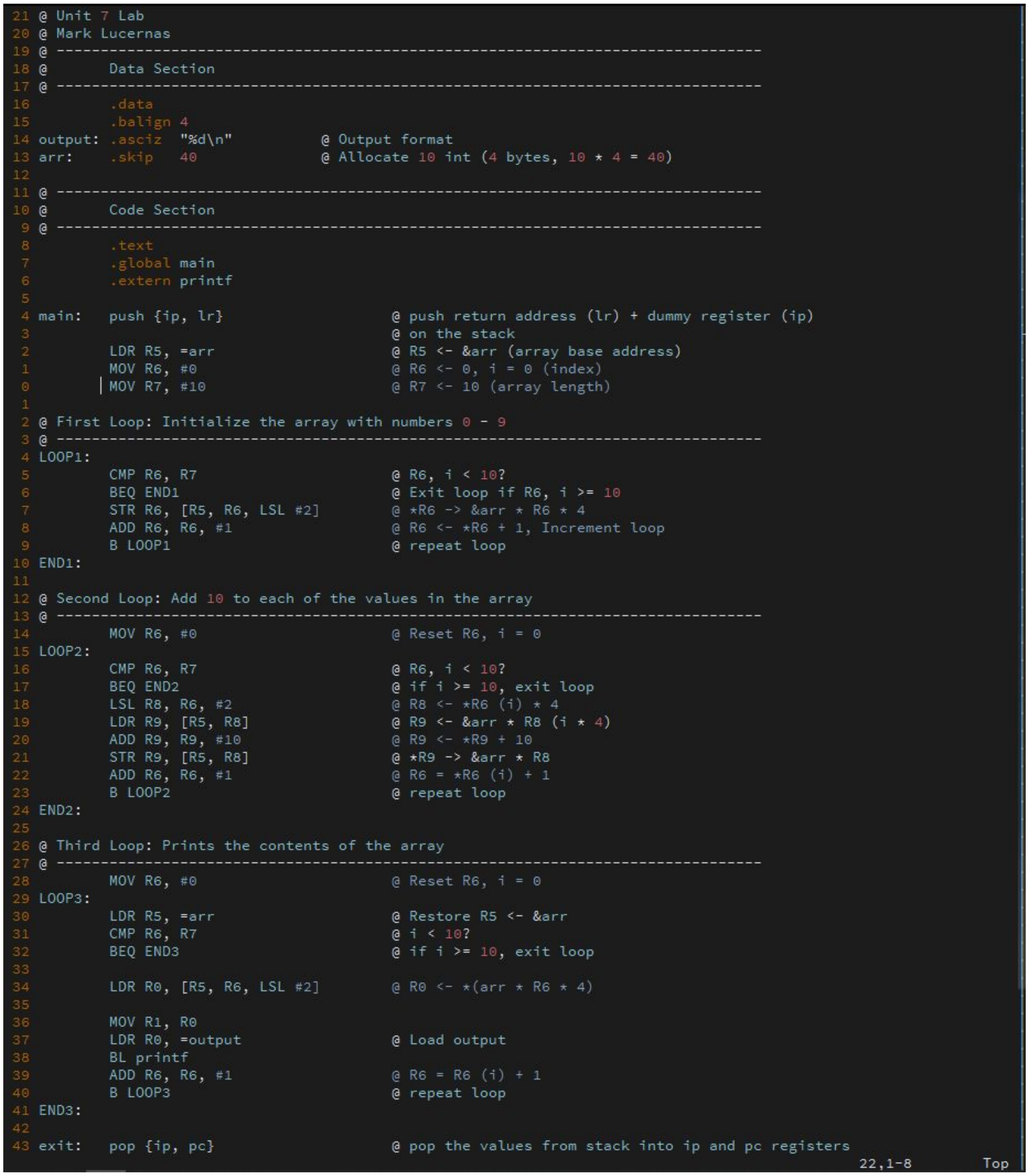
```
+--Register group: general--------------------------------------------------------------+
|r0             0x1      1                              r1             0x7efff284      2130702980
|r2             0x7efff28c       2130702988             r3             0x10440  66624
|r4             0x104bc  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)
```

Compile & Program Execution Screen Capture



```
cisc-211-13@raspberrypi:~/unit-7_lab $ make
as -o DDCA_unit-7.o DDCA_unit-7.s
gcc -o DDCA_unit-7 DDCA_unit-7.o
cisc-211-13@raspberrypi:~/unit-7_lab $ ./DDCA_unit-7
10
11
12
13
14
15
16
17
18
19
cisc-211-13@raspberrypi:~/unit-7_lab $
```

Assembly Source Code Screen Capture

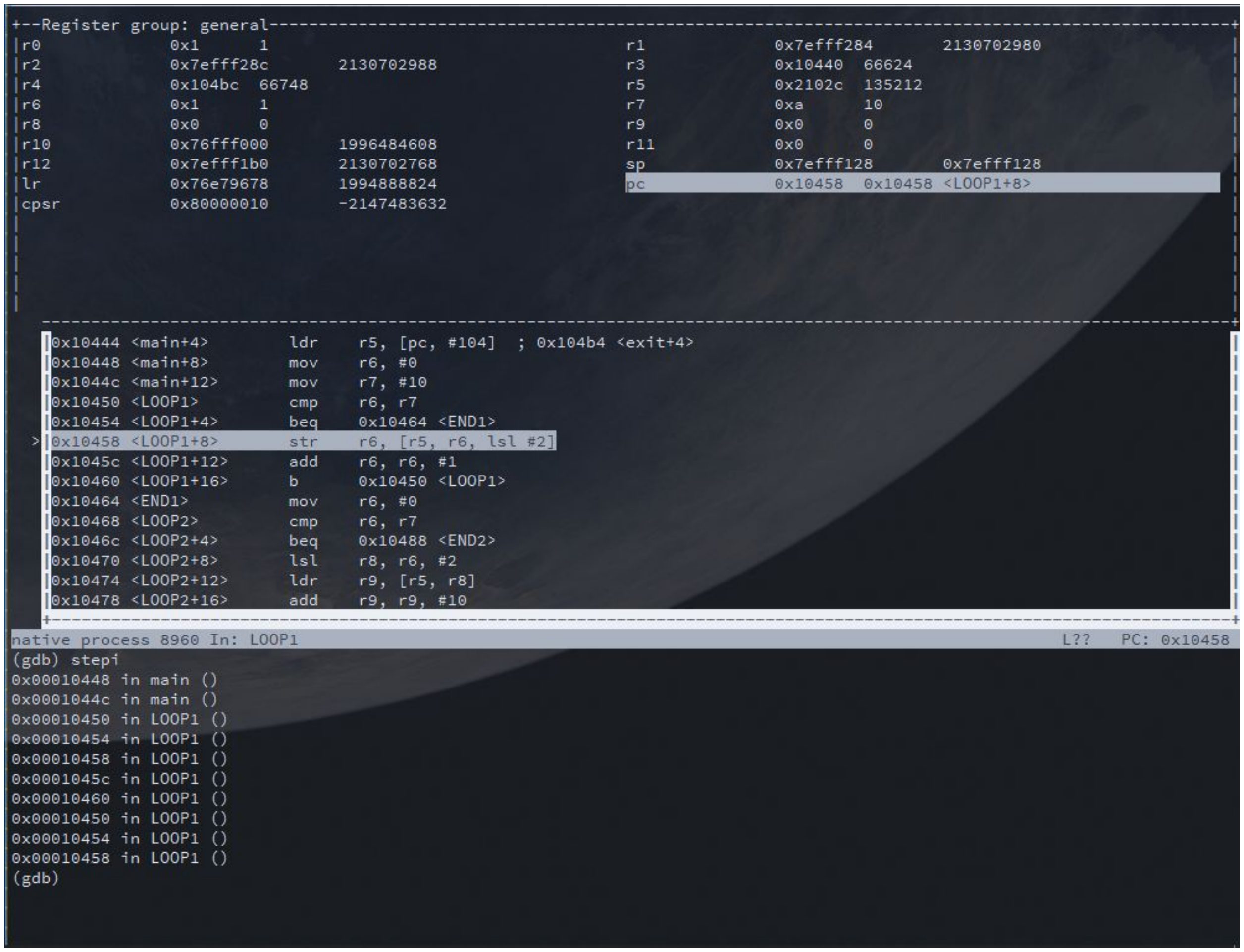


```
@ Unit 7 Lab
@ Mark Lucernas
@ --------------------------------------------------------------------------------
@       Data Section
@ --------------------------------------------------------------------------------
        .data
        .balign 4
output: .asciz  "%d\n"          @ Output format
arr:    .skip   40              @ Allocate 10 int (4 bytes, 10 * 4 = 40)

@ --------------------------------------------------------------------------------
@       Code Section
@ --------------------------------------------------------------------------------
        .text
        .global main
        .extern printf

main:   push {ip, lr}                   @ push return address (lr) + dummy register (ip)
                                        @ on the stack
        LDR R5, =arr                    @ R5 <- &arr (array base address)
        MOV R6, #0                      @ R6 <- 0, i = 0 (index)
        MOV R7, #10                     @ R7 <- 10 (array length)

@ First Loop: Initialize the array with numbers 0 - 9
@ --------------------------------------------------------------------------------
LOOP1:
        CMP R6, R7                      @ R6, i < 10?
        BEQ END1                        @ Exit loop if R6, i >= 10
        STR R6, [R5, R6, LSL #2]        @ *R6 -> &arr * R6 * 4
        ADD R6, R6, #1                  @ R6 <- *R6 + 1, Increment loop
        B LOOP1                         @ repeat loop
END1:

@ Second Loop: Add 10 to each of the values in the array
@ --------------------------------------------------------------------------------
        MOV R6, #0                      @ Reset R6, i = 0
LOOP2:
        CMP R6, R7                      @ R6, i < 10?
        BEQ END2                        @ if i >= 10, exit loop
        LSL R8, R6, #2                  @ R8 <- *R6 (i) * 4
        LDR R9, [R5, R8]                @ R9 <- &arr * R8 (i * 4)
        ADD R9, R9, #10                 @ R9 <- *R9 + 10
        STR R9, [R5, R8]                @ *R9 -> &arr * R8
        ADD R6, R6, #1                  @ R6 = *R6 (i) + 1
        B LOOP2                         @ repeat loop
END2:

@ Third Loop: Prints the contents of the array
@ --------------------------------------------------------------------------------
        MOV R6, #0                      @ Reset R6, i = 0
LOOP3:
        LDR R5, =arr                    @ Restore R5 <- &arr
        CMP R6, R7                      @ i < 10?
        BEQ END3                        @ if i >= 10, exit loop

        LDR R0, [R5, R6, LSL #2]        @ R0 <- *(arr * R6 * 4)

        MOV R1, R0
        LDR R0, =output                 @ Load output
        BL printf
        ADD R6, R6, #1                  @ R6 = R6 (i) + 1
        B LOOP3                         @ repeat loop
END3:

exit:   pop {ip, pc}                    @ pop the values from stack into ip and pc registers
                                                                        22,1-8        Top
```

DDCA_unit-7.s

```
@ Unit 7 Lab
@ Mark Lucernas
@ ---------------------------------------------------------------------------
@       Data Section
@ ---------------------------------------------------------------------------
        .data
        .balign 4
output: .asciz  "%d\n"          @ Output format
arr:    .skip   40              @ Allocate 10 int (4 bytes, 10 * 4 = 40)

@ ---------------------------------------------------------------------------
@       Code Section
@ ---------------------------------------------------------------------------
        .text
        .global main
        .extern printf

main:   push {ip, lr}                   @ push return address (lr) + dummy register (ip)
                                        @ on the stack
        LDR R5, =arr                    @ R5 <- &arr (array base address)
        MOV R6, #0                      @ R6 <- 0, i = 0 (index)
        MOV R7, #10                     @ R7 <- 10 (array length)

@ First Loop: Initialize the array with numbers 0 - 9
@ ---------------------------------------------------------------------------
LOOP1:
        CMP R6, R7                      @ R6, i < 10?
        BEQ END1                        @ Exit loop if R6, i >= 10
        STR R6, [R5, R6, LSL #2]        @ *R6 -> &arr * R6 * 4
        ADD R6, R6, #1                  @ R6 <- *R6 + 1, Increment loop
        B LOOP1                         @ repeat loop
END1:

@ Second Loop: Add 10 to each of the values in the array
@ ----------------------------------------------------------------------------
        MOV R6, #0                  @ Reset R6, i = 0
LOOP2:
        CMP R6, R7                  @ R6, i < 10?
        BEQ END2                    @ if i >= 10, exit loop
        LSL R8, R6, #2              @ R8 <- *R6 (i) * 4
        LDR R9, [R5, R8]            @ R9 <- &arr * R8 (i * 4)
        ADD R9, R9, #10             @ R9 <- *R9 + 10
        STR R9, [R5, R8]            @ *R9 -> &arr * R8
        ADD R6, R6, #1              @ R6 = *R6 (i) + 1
        B LOOP2                     @ repeat loop
END2:

@ Third Loop: Prints the contents of the array
@ ----------------------------------------------------------------------------
        MOV R6, #0                  @ Reset R6, i = 0
LOOP3:
        LDR R5, =arr                @ Restore R5 <- &arr
        CMP R6, R7                  @ i < 10?
        BEQ END3                    @ if i >= 10, exit loop

        LDR R0, [R5, R6, LSL #2]    @ R0 <- *(arr * R6 * 4)

        MOV R1, R0
        LDR R0, =output             @ Load output
        BL printf
        ADD R6, R6, #1              @ R6 = R6 (i) + 1
        B LOOP3                     @ repeat loop
END3:

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

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)
```