

cisc-211-13@raspberrypi

Unit 7 Lab

Overview Screen Capture

```

1 0 main: MARK:XP515 =
2 0 Unit 7 Lab
3 0 Mark Lucernas
4 -----
5 0 Data Section
6 0 -----
7 0 .data
8 0 .balign 4
9 0 output: .asciz "Mdn"          @ Output format
10 0 arr: .skip 40                @ Allocate 10 int (4 bytes, 10 * 4 = 40)
11 0
12 0
13 0 Code Section
14 0 -----
15 0 .text
16 0 .global main
17 0 .extern printf
18 0
19 0
20 0 main: push {ip, lr}           @ push return address (lr) & dummy register (ip)
21 0                                @ on the stack
22 0 LDR R5, =arr                 @ R5 <- &arr (array base address)
23 0 MOV R6, #0                  @ R6 <- 0, i = 0 (index)
24 0 MOV RT, #10                 @ RT <- 10 (array length)
25 0
26 0 First Loop: Initialize the array with numbers 0 - 9
27 0
28 0 LOOP1:
29 0     CMP R6, RT               @ R6, i < 10?
30 0     BEQ END1                @ Exit loop if R6, i == 10
31 0     STR R6, [R5, RSL #2]     @ R6 -> &arr + R6 * 4
32 0     ADD R6, R6, #1          @ R6 <- R6 + 1, Increment loop
33 0     B LOOP1                 @ repeat loop
34 0 END1:
35 0
36 0 Second Loop: Add 10 to each of the values in the array
37 0
38 0 MOV R6, #0                  @ Reset R6, i = 0
39 0
40 0 LOOP2:
41 0     CMP R6, RT               @ R6, i < 10?
42 0     BEQ END2                @ If i == 10, exit loop
43 0     LSL R6, R6, #2           @ R6 <- R6 (i) * 4
44 0     LDR R9, [R5, R6]         @ R9 <- &arr + R6 (i * 4)
45 0     ADD R9, R9, #10          @ R9 <- R9 + 10
46 0     STR R9, [R5, R6]         @ R9 -> &arr + R6
47 0     ADD R6, R6, #1          @ R6 = R6 (i) + 1
48 0     B LOOP2                 @ repeat loop
49 0 END2:
50 0
51 0 Third Loop: Prints the contents of the array
52 0
53 0 MOV R6, #0                  @ Reset R6, i = 0
54 0
55 0 LOOP3:
56 0     LDR R5, =arr             @ Restore R5 <- &arr
57 0     CMP R6, RT               @ i < 10?
58 0     BEQ END3                @ If i == 10, exit loop
59 0     LDR R9, [R5, R6, LSL #2] @ R9 <- &arr + R6 * 4
60 0     MOV R1, R9               @ Load output
61 0     LDR R9, =output          @ Load output
62 0     BL printf                @ printf
63 0     ADD R6, R6, #1          @ R6 = R6 (i) + 1
64 0     B LOOP3                 @ repeat loop
65 0 END3:
66 0
67 0 exit: pop {ip, pc}           @ pop the values from stack into ip and pc registers
68 0
69 0
70 0

```

```

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      2130702988  r1      0x7efff284  2130702988
r2      0x7efff28c 2130702988  r3      0x10440 66624
r4      0x104bc 66748  r5      0x2102c 155212
r6      0x1      0      r7      0xa 10
r8      0x0 0      r9      0x0 0
r10     0x7efff008 1996484688 r11     0x0 0
r12     0x7efff100 2130702988 r13     0x7efff128 0x7efff128
r14     0x7eff7e78 1994888424 sp      0x10458 0x10458 <LOOP1+8>
r15     0x0 0
cpsr    0x00000010 -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> rfi r0, r5, r6, lsl #2
0x10460 <LOOP1+12> add r6, r6, #1
0x10464 <LOOP1+16> b 0x10468 <LOOP1>
0x10468 <END1> mov r6, #0
0x1046c <END1> mov r6, #0
0x10470 <END1> cmp r6, r7
0x10474 <LOOP2+4> beq 0x10478 <END2>
0x10478 <LOOP2+8> lsl r6, r6, #2
0x1047c <LOOP2+12> ldr r9, [r5, r6]
0x10480 <LOOP2+16> add r9, r9, #10

init: process #966 In: LOOP1
(gdb) step
0x0010440 in main ()
0x001044c in main ()
0x0010450 in LOOP1 ()
0x0010454 in LOOP1 ()
0x0010458 in LOOP1 ()
0x001045c in LOOP1 ()
0x0010460 in LOOP1 ()
0x0010464 in LOOP1 ()
0x0010468 in LOOP1 ()
0x001046c in LOOP1 ()
0x0010470 in LOOP1 ()
0x0010474 in LOOP1 ()
0x0010478 in LOOP1 ()
0x0010480 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

```
21 @ Unit 7 Lab
20 @ Mark Lucernas
19 @ -----
18 @      Data Section
17 @ -----
16      .data
15      .balign 4
14 output: .asciz "%d\n"      @ Output format
13 arr:     .skip 40           @ Allocate 10 int (4 bytes, 10 * 4 = 40)
12
11 @ -----
10 @      Code Section
9 @ -----
8      .text
7      .global main
6      .extern printf
5
4 main:  push {ip, lr}          @ push return address (lr) + dummy register (ip)
3              @ on the stack
2      LDR R5, =arr             @ R5 <- &arr (array base address)
1      MOV R6, #0               @ R6 <- 0, i = 0 (index)
0      | MOV R7, #10            @ R7 <- 10 (array length)
1
2 @ First Loop: Initialize the array with numbers 0 - 9
3 @ -----
4 LOOP1:
5      CMP R6, R7               @ R6, i < 10?
6      BEQ END1                @ Exit loop if R6, i >= 10
7      STR R6, [R5, R6, LSL #2] @ *R6 -> &arr * R6 * 4
8      ADD R6, R6, #1           @ R6 <- *R6 + 1, Increment loop
9      B LOOP1                 @ repeat loop
10 END1:
11
12 @ Second Loop: Add 10 to each of the values in the array
13 @ -----
14      MOV R6, #0               @ Reset R6, i = 0
15 LOOP2:
16      CMP R6, R7               @ R6, i < 10?
17      BEQ END2                @ if i >= 10, exit loop
18      LSL R8, R6, #2           @ R8 <- *R6 (i) * 4
19      LDR R9, [R5, R8]         @ R9 <- &arr * R8 (i * 4)
20      ADD R9, R9, #10          @ R9 <- *R9 + 10
21      STR R9, [R5, R8]         @ *R9 -> &arr * R8
22      ADD R6, R6, #1           @ R6 = *R6 (i) + 1
23      B LOOP2                 @ repeat loop
24 END2:
25
26 @ Third Loop: Prints the contents of the array
27 @ -----
28      MOV R6, #0               @ Reset R6, i = 0
29 LOOP3:
30      LDR R5, =arr             @ Restore R5 <- &arr
31      CMP R6, R7               @ i < 10?
32      BEQ END3                @ if i >= 10, exit loop
33
34      LDR R0, [R5, R6, LSL #2] @ R0 <- *(arr * R6 * 4)
35
36      MOV R1, R0
37      LDR R0, =output           @ Load output
38      BL printf
39      ADD R6, R6, #1           @ R6 = R6 (i) + 1
40      B LOOP3                 @ repeat loop
41 END3:
42
43 exit:  pop {ip, pc}           @ pop the values from stack into ip and pc registers
```

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
r2      0x7efff28c 2130702988
r4      0x104bc  66748
r6      0x1      1
r8      0x0      0
r10     0x76fff000 1996484608
r12     0x7efff1b0 2130702768
lr      0x76e79678 1994888824
cpsr    0x80000010 -2147483632

r1      0x7efff284 2130702980
r3      0x10440  66624
r5      0x2102c  135212
r7      0xa      10
r9      0x0      0
r11     0x0      0
sp      0x7efff128 0x7efff128
pc      0x10458 0x10458 <LOOP1+8>

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)
```