

## Lab 10 – Computer Systems (Mohamed Shafi Uzman Fassy - 102608927)

### 10.1

The screenshot displays a computer system simulator interface. The main window is titled "Program" and contains assembly code. The code defines a "flash" routine that calls "drawpixel" and "delay". The "drawpixel" routine pushes the current LR (Link Register) to the stack, calls "delay", and then pops the LR. The "delay" routine pushes R3, R4, R5, and R6 to the stack, updates the time, and compares the elapsed time against a delay value. The "timer" routine updates the time and compares it against the delay value. The "drawpixel" routine is called from the "flash" routine. The processor status is shown on the right, including PC, LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, and R0. The Count is 122871880. The Current Instruction is not shown. The Status bits are NZCV 0000. The Input/Output section shows a red square on a black background.

```
1  mov R2, #1      ; 1sec delay time
2 flash:
3  mov R0, #.red
4  mov R1, R2
5  bl drawpixel    ; flashes on
6  mov R0, #.white
7  mov R1, R2
8  bl drawpixel    ; flashes off
9  b flash
10 halt
11 drawpixel:
12  push {R3,R4}
13  mov R3, R0      ; copy pixel color to R3
14  mov R4, R1      ; copy delay time to R4
15  str R3, [R0, #Pixel367] ; draw pixel with color
16  push {R0, LR}   ; backup R0 and LR before param pass and function call
17  mov R0, R4      ; pass delay time to delay function
18  bl delay        ; call delay function
19  pop {R0, LR}    ; restore R0 and LR after function call
20  pop {R3,R4}
21  ret
22 delay:
23  push {R3,R4,R5,R6}
24  mov R3, R0      ; move delay time param into R3
25  ldr R4, .Time   ; get start time
26 timer:
27  ldr R5, .Time   ; update time
28  sub R6, R5, R4  ; calc elapsed time
29  cmp R6, R3      ; compare elapsed to delay time
30  blt timer
31  pop {R3,R4,R5,R6}
32  ret
```

c) we use LR to push it into the stack so that after the nesting function is completed it will know the address to return back to and execute the next instruction.

### 10.2

The screenshot displays a computer system simulator interface. The main window is titled "Program" and contains assembly code. The code defines a "flash" routine that calls "drawpixel" and "delay". The "drawpixel" routine pushes the current LR (Link Register) to the stack, calls "delay", and then pops the LR. The "delay" routine pushes R3, R4, R5, and R6 to the stack, updates the time, and compares the elapsed time against a delay value. The "timer" routine updates the time and compares it against the delay value. The "drawpixel" routine is called from the "flash" routine. The processor status is shown on the right, including PC, LR, SP, R12, R11, R10, R9, R8, R7, R6, R5, R4, R3, R2, R1, and R0. The Count is 23547792. The Current Instruction is not shown. The Status bits are NZCV 0000. The Input/Output section shows a red square on a black background.

```
1  mov R2, #1      ; 1sec delay time
2 flash:
3  mov R0, #.red
4  mov R1, R2
5  bl drawpixel    ; flashes on
6  mov R0, #.white
7  mov R1, R2
8  bl drawpixel    ; flashes off
9  add R3, R3, #1
10 cmp R3, #3
11 bne flash
12 mov R0, #2
13 bl delaytwosec
14 b flash
15 halt
16 drawpixel:
17  push {R3,R4}
18  mov R3, R0      ; copy pixel color to R3
19  mov R4, R1      ; copy delay time to R4
20  str R3, [R0, #Pixel367] ; draw pixel with color
21  push {R0, LR}   ; backup R0 and LR before param pass and function call
22  mov R0, R4      ; pass delay time to delay function
23  bl delayonesec  ; call delay function
24  pop {R0, LR}    ; restore R0 and LR after function call
25  pop {R3,R4}
26  ret            ; job done
27 delayonesec:
28  push {R3,R4,R5,R6}
29  mov R3, R0      ; move delay time param into R3
30  ldr R4, .Time   ; get start time
31 timer:
32  ldr R5, .Time   ; update time
33  sub R6, R5, R4  ; calc elapsed time
34  cmp R6, R3      ; compare elapsed to delay time
35  blt timer
36  pop {R3,R4,R5,R6}
37  ret
38 delaytwosec:
39  push {R3,R4,R5,R6}
40  mov R3, R0      ; move delay time param into R3
```

## 10.3

### Program

```

23 | MOV R0, R4      ; pass delay time to delay function
24 | BL delayonesec ; call delay function
25 | Pop {R0, LR}   ; restore R0 and LR after function call
26 | Pop {R3,R4}
27 | RET           ; job done
28 | delayonesec:
29 | push {R3,R4,R5,R6}
30 | MOV R3, R0     ; move delay time param into R3
31 | LDR R4, .Time  ; get start time
32 | timer:
33 | LDR R5, .Time  ; update time
34 | SUB R6, R5, R4 ; calc elapsed time
35 | CMP R6, R3     ; compare elapsed to delay time
36 | BLT timer
37 | pop {R3,R4,R5,R6}
38 | RET
39 | delaytwosec:
40 | push {R3,R4,R5,R6}
41 | MOV R3, R0     ; move delay time param into R3
42 | LDR R4, .Time  ; get start time
43 | timertwo:
44 | LDR R5, .Time  ; update time
45 | SUB R6, R5, R4 ; calc elapsed time
46 | CMP R6, R3     ; compare elapsed to delay time
47 | BLT timertwo
48 | pop {R3,R4,R5,R6}
49 | MOV R3, #0
50 | RET
51 | flashpattern:
52 | PUSH {R3, R4}
53 | MOV R3, #t1
54 | STR R3, .WriteString
55 | LDR R6, .InputNum
56 | MOV R4, #t2
57 | STR R4, .WriteString
58 | LDR R7, .InputNum
59 | POP {R3, R4}
60 | RET
61 | t1: .ASCIIZ "Enter the number of rapid 1 second flashes before the pause:\n"
62 | t2: .ASCIIZ "Enter the pause time (in seconds) between each set of rapid flashes:\n"

```

### Processor

PC	0x000000c0
LR	0x00000008
SP	0x000ffff8
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000000
R6	0x00000000
R5	0x00000000
R4	0x00000000
R3	0x000000d4
R2	0x00000001
R1	0x00000000
R0	0x00000000

Count:

Current Instruction:

Status bits: NZCV 0000

### Input/Output

Enter the number of rapid 1 second flashes before the pause:

### Program

```

1 | mov R2, #1      ; 1sec delay time
2 | BL flashpattern
3 | flash:
4 | MOV R0, #.red
5 | MOV R1, R2
6 | BL drawpixel   ; flashes on
7 | MOV R0, #.white
8 | MOV R1, R2
9 | BL drawpixel   ; flashes off
10 | ADD R3, R3, #1
11 | CMP R3, R6     ;dictate number of rapid flashes before the big delay
12 | BNE flash
13 | MOV R0, R7     ;dictate delay second of the pause after rapid flashes
14 | BL delaytwosec
15 | B flash
16 | HALT
17 | drawpixel:
18 | PUSH {R3,R4}
19 | MOV R3, R0     ; copy pixel colour to R3
20 | MOV R4, R1     ; copy delay time to R4
21 | STR R3, .Pixel367 ; draw pixel with colour
22 | PUSH {R0, LR}  ; backup R0 and LR before param pass and function call
23 | MOV R0, R4     ; pass delay time to delay function
24 | BL delayonesec ; call delay function
25 | Pop {R0, LR}   ; restore R0 and LR after function call
26 | Pop {R3,R4}
27 | RET           ; job done
28 | delayonesec:
29 | push {R3,R4,R5,R6}
30 | MOV R3, R0     ; move delay time param into R3
31 | LDR R4, .Time  ; get start time
32 | timer:
33 | LDR R5, .Time  ; update time
34 | SUB R6, R5, R4 ; calc elapsed time
35 | CMP R6, R3     ; compare elapsed to delay time
36 | BLT timer
37 | pop {R3,R4,R5,R6}
38 | RET
39 | delaytwosec:
40 | push {R3,R4,R5,R6}

```

### Processor

PC	0x000000cc
LR	0x00000008
SP	0x000ffff8
R12	0x00000000
R11	0x00000000
R10	0x00000000
R9	0x00000000
R8	0x00000000
R7	0x00000002
R6	0x00000003
R5	0x00000000
R4	0x00000112
R3	0x000000d4
R2	0x00000001
R1	0x00000000
R0	0x00000000

Count:

Current Instruction:

Status bits: NZCV 0000

### Input/Output

flashes before the pause:

Enter the pause time (in seconds) between each set of rapid flashes: