Trac Duc Anh Luong - 103488117

Lab 9

4. Provide a brief description of the following:

4.1. What is the code in each file doing?

- kernel7.ASM
  Calculate the factorial of a number and flash the LED from GPIO 18 the number of times of the result calculated above.
- TIMER.ASM
  Loads the r2 register from the previous code with a value, subtracts one from it, and compares it to zero. If the number reaches zero, the program will terminate; otherwise, the process will be repeated until the number reaches zero.
- FACTORIAL.ASM
  Takes the values 4 in registers r0 and r1 from kernel7.ASM and subtracts one from r1 before comparing it to one; if it is true, it will stop; otherwise, multiply r0 and r1 and store the result in r0, which was 3(r1)*4(r0) in the first case. As a result, if we repeat it until r1 becomes one, we will end up with 4! in r0.

4.2. What register holds the input to the program (and what is the input)?

Register r0 holds the input to the program, which is #4.

5.

Stage 1: Open Kernel7.asm and separate out GPIO function for initialising LED

Code

```
;Calculate
mov r1,#4 ;input
mov sp,$1000 ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer

BASE = $3F000000 ;RP2 and RP3

;GPIO_SETUP
GPIO_OFFSET = $200000
mov r0,BASE
bl SETUP_LED

loop$:
 mov r1,#1
 lsl r1,#18
 str r1,[r0,#28] ;turn LED on
```

```
 mov r2,$0F0000 ;not using r2 for anything else so no need to
push/pop
 bl TIMER
 mov r1,#1
 lsl r1,#18
 str r1,[r0,#40] ;turn LED off
 mov r2,$0F0000
 bl TIMER
sub r7,#1
cmp r7,#0
bne loop$ ;end of outer loop. Runs r7 times

wait:
b wait

include "TIMER.asm"
include "factorialj.asm"

SETUP_LED:
;param r0=BASE
orr r0,GPIO_OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 to output
bx lr
```

Output

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm Kernel7.asm
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
2 passes, 152 bytes.
```

2. Separate out GPIO function for Flashing LED

Code

```
;Calculate
mov r1,#4 ;input
mov sp,$1000 ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer

BASE = $3F000000 ;RP2 and RP3

;GPIO_SETUP
```

```
GPIO_OFFSET = $200000
mov r0,BASE
bl SETUP_LED

push {r0,r1}
mov r0,BASE
mov r1,r7
bl FLASH
pop {r0,r1}

wait:
b wait

include "TIMER.asm"
include "factorialj.asm"

SETUP_LED:
;param r0=BASE
orr r0,GPIO_OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 to output
bx lr

FLASH:
;param r0=BASE
;param r1 = number of flashes
orr r0,GPIO_OFFSET
mov r7,r1

loop$:
 mov r1,#1
 lsl r1,#18
 str r1,[r0,#28] ;turn LED on
 mov r2,$0F0000 ;not using r2 for anything else so no need to
push/pop
 bl TIMER
 mov r1,#1
 lsl r1,#18
 str r1,[r0,#40] ;turn LED off
 mov r2,$0F0000
 bl TIMER
sub r7,#1
cmp r7,#0
bne loop$ ;end of outer loop. Runs r7 times
```

```
bx lr
```

Output

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm Kernel7.asm
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
2 passes, 184 bytes.
```

(compile passes but the LED flashes > 24 times)

Stage 3: Move GPIO functions into their own file

Code

```
;Calculate
mov r1,#4 ;input
mov sp,$1000 ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer

BASE = $3F000000 ;RP2 and RP3 ;GPIO_SETUP

mov r0,BASE
bl SETUP_LED
mov r0,BASE
bl SETUP_LED
mov r0,BASE
mov r1,r7
bl FLASH

wait:
b wait

include "TIMER.asm"
include "factorialj.asm"
include "GPIO.asm"
```

Output

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm Kernel7.asm
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
2 passes, 200 bytes.
```

Level Up

Code

```
;Kernel7.asm
mov r1,#4 ;input
mov sp,$1000 ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer

BASE = $FE000000 ;RP4
;GPIO_SETUP
mov r0,BASE

bl SETUP_LED

mov r0,BASE
mov r1,r7
bl FLASH

wait:
b wait
include "timer2_2Param.asm"
include "factorialj.asm"
include "GPIO.asm"

;GPIO.asm
GPIO_OFFSET = $200000
SETUP_LED:
;param r0=BASE
orr r0,GPIO_OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 to output
bx lr

FLASH:
;param r0=BASE
;need BASE for timer, so copy to r2 here
mov r2,r0
;param r1 = number of flashes
orr r0,GPIO_OFFSET
mov r7,r1
loop$:
 mov r1,#1
 lsl r1,#18
 str r1,[r0,#28] ;turn LED on
  push {r0,r1,r7,lr} ;r0,r1,r7 in use push and then set parameters
```

```
  mov r0,BASE
  mov r1,$0F0000
  bl Delay
  pop {r0,r1,r7,lr}
  mov r1,#1
  lsl r1,#18
  str r1,[r0,#40] ;turn LED off
  push {r0,r1,r7,lr} ;r0,r1,r7 in use push and then set parameters
  mov r0,BASE
  mov r1,$0F0000
  bl Delay
  pop {r0,r1,r7,lr}
sub r7,r7,#1
cmp r7,#0
bgt loop$ ;end of outer loop. Runs r7 times
bx lr
```

Output

```
ducanh DESKTOP-RN4OP9Q ~ Downloads FASMARM $ ./fasmarm Kernel7.asm
flat assembler for ARM  version 1.43 (built on fasm 1.73.02)  (16384 kilobytes memory)
2 passes, 228 bytes.
```