# How to separate out functions in Assembly code.

COS10004 CS Lab 9

# Open Kernel7.asm and separate out GPIO function for initialising LED

```
;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 3; GPIO SETUP
GPIO OFFSET = $200000
mov r0,BASE
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 to output
loop$:
 mov r1,#1
 lsl r1,#15
  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,#15
  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"
```

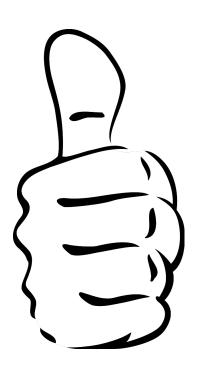
# Open Kernel7.asm and separate out GPIO function for initialising LED

```
;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 3 ; GPIO SETUP
GPIO OFFSET = $200000
mov r0,BASE
bl SETUP LED
SETUP LED:
;param r0=BASE
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4] ;set GPIO18 to output
bx lr
loop$:
 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"
```

# Open Kernel7.asm and separate out GPIO function for initialising LED

```
;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 ; 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 fall 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 .7 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
```





```
Separate out GPIO function for Flashing LED
;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 ; 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
```

```
Separate out GPIO function for Flashing LED
;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 ; GPIO SETUP
GPIO OFFSET = $200000
mov r0,BASE
bl SETUP LED
mov r0,BASE
mov r1,r7
bl FLASH
FLASH:
;param r0=BASE
;param r1 = number of flashes
orr r0,GPIO OFFSET
mov r7,r1
loop$:
 mov r1,#1
 lsl r1,#18
  str r1,[r0,#28]
                 ;turn LED on
                 ;not using r2 for anything else so no need to push/pop
 mov r2,$0F0000
 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 1r
wait:
b wait
include "TIMER.asm"
include "factorialj.asm"
```

```
GPIO OFFSET = $200000
                                 Separate out GPIO function for Flashing LED
mov r0,BASE
bl SETUP LED
mov r0,BASE
mov r1,r7
bl FLASH
wait:
b wait
include "TIMER.asm"
include "factorialj.
SETUP LED:
;param r0=BASE
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4]
                   ;set GPIO18 to ou put
bx lr
FLASH:
;param r0=BASE
;param r1 = number of flashes
orr r0,GPIO_OFFSET
mov r7,r1
loop$:
 mov r1,#1
 lsl r1,#18
  str r1,[r0,#28]
                  ;turn LED on
                 ;not using r2 for anything else so no need to push/pop
 mov r2,$0F0000
 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
```







- Problem. Flashes for more than 24 times.
- Why?
  - this is a challenge to work out, but follow the logic and see if you can, then proceed to fix it.

### Why...

- The link register has been overwritten by the nested function call: ;sequence:
- 1. call FACTORIAL (from factorialj.asm) overwrites lr
- 2. returns 4! (24) (copies lr to pc)
- 3. put 24 in r7
- 4. call SETUP\_LED (from GPIO.asm) overwrites lr
- 5. returns (copies Ir to pc)
- 6. call FLASH (from GPIO.asm) overwrites lr
- 7. FLASH reads value in r0 (4!)
- 8. FLASH calls TIMER (from timer2\_2param.asm) overwrites lr and repeats 24 times (copies lr to pc)
- 9. FLASH returns (copies Ir to pc)
  - wrong Ir correct one has been overwritten
- 10. return and run infinite loop

### Solution

Edit the FLASH function to backup the value in "Ir" onto the stack before TIMER is called (and to restore from the backup after TIMER completes).

```
So:
bl TIMER
should be replaced with:
push {lr}
bl TIMER
pop {lr}"
```

```
GPIO OFFSET = $200000
                                 Separate out GPIO function for Flashing LED
mov r0,BASE
bl SETUP LED
mov r0,BASE
mov r1, r7
bl FLASH
wait:
b wait
include "TIMER.asm"
include "factorialj.asm"
SETUP LED:
;param r0=BASE
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4]
                 ;set GPIO18 to output
bx lr
FLASH:
;param r0=BASE
;param r1 = number of flashes
orr r0,GPIO OFFSET
mov r7, r1
loop$:
 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
    push {lr}
    bl TIMER
    pop {lr}
                                                                                      these fixed it.
 mov r1,#1
  lsl r1,#18
                                                                                      ASM is weird.
  str r1,[r0,#40]
                  ;turn LED off
 mov r2,$0F0000
    push {lr}
    bl TIMER
    pop {lr}
sub r7,#1
cmp r7,#0
bne loop$
           ;end of outer loop. Runs r7 times
```

bx lr

#### 3. Move GPIO functions into their own file

```
: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 ;GPIO SETUP
GPIO OFFSET = $200000
mov r0, BASE
bl SETUP LED
GPIO OFFSET = $200000
mov r0,BASE
bl SETUP LED
mov r0, BASE
mov r1, r7
bl FLASH
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
     push {lr}
     bl TIMER
    pop {lr}
 mov r1,#1
  lsl r1,#18
  str r1,[r0,#40] ;turn LED off
  mov r2,$0F0000
    push {lr}
     bl TIMER
     pop {lr}
sub r7, r7, #1
cmp r7,#0
ble loop$ ;end of outer loop. Runs r7 times
bx 1r
```

#### 3. Move GPIO functions into their own file

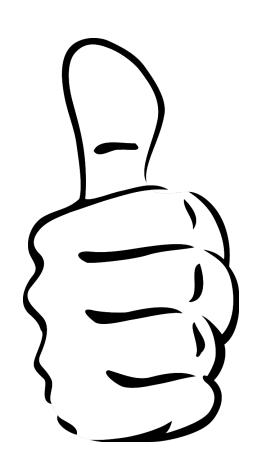
bat loop\$ :end of outer loop. Runs r7 times

```
;kernel7.asm
                                                  ;param r0=BASE
:Calculate
                                                 orr r0,GPIO OFFSET
mov r1,#4 ;input
                                                 mov r1,#1
mov sp,$1000 ; make room on the stack
                                                 lsl r1,#24
mov r0,r1
                                                 str r1,[r0,#4] ;set GPIO18 to output
bl FACTORIAL
                                                 bx lr
mov r7,r0 ;store answer
BASE = $3F000000 ;RP2 ;GPIO SETUP
                                                 FLASH:
GPIO OFFSET = $200000
                                                 ;param r0=BASE
mov r0,BASE
                                                  ;param r1 = number of flashes
                                                 orr r0,GPIO OFFSET
bl SETUP LED
GPIO OFFSET = $200000
                                                 mov r7,r1
mov r0,BASE
                                                 loop$:
                                                   mov r1,#1
bl SETUP LED
                                                   lsl r1,#18
mov r0, BASE
mov r1,r7
                                                   str r1,[r0,#28] ;turn LED on
                                                   mov r2,$0F0000 ;not using r2 for anything
bl FLASH
                                                 else so no need to push/pop
wait:
                                                    push {lr}
b wait
                                                      bl TIMER
                                                      pop {lr}
include "TIMER.asm"
                                                   mov r1,#1
include "factorialj.asm"
                                                   lsl r1,#18
include "GPIO.asm"
                                                   str r1,[r0,#40] ;turn LED off
                                                   mov r2,$0F0000
;GPIO.asm
                                                    push {lr}
SETUP LED:
                                                      bl TIMER
                                                      pop {lr}
                                                 sub r7, r7, #1
                                                 cmp r7,#0
```

#### Move GPIO functions into their own file

```
:kernel7.asm
: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; GPIO SETUP
mov r0,BASE
bl SETUP LED
mov r0,BASE
bl SETUP LED
                    We don't actually use
mov r0,BASE
mov r1,r7
                        GPIO OFFSET in
bl FLASH
                   kernel7, so we'll move it
wait:
                      to GPIO where it is
b wait
                            relevant.
include "TIMER.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
;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
   push {lr}
     bl TIMER
     pop {lr}
  mov r1,#1
  lsl r1,#18
  str r1,[r0,#40]
                  turn LED off;
  mov r2,$0F0000
   push {lr}
     bl TIMER
     pop {lr}
sub r7, r7, #1
cmp r7.#0
```





### Level Up

- Suppose your function needs lots of registers.
- But you're already using them.
- push them onto the stack
- set param registers
- call function
- pop registers
- Try this replace the dumb timer in Lab 8 with the accurate timer discussed in Week 9's lecture.

### Step 1. Substitute timer function

#### ;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 = \$3F0000000; RP2 and RP3
;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"
```

```
; timer2 2Param.asm
```

```
Delay: ;this function has 2 parameters

TIMER_OFFSET=$3000

mov r3,r0 ;BASE - depends on Pi model

orr r3,TIMER_OFFSET

mov r4,r1 ;$80000 passed as a parameter

ldrd r6,r7,[r3,#4]

mov r5,r6

loopt1: ;label still has to be

different from one in _start

ldrd r6,r7,[r3,#4]

sub r8,r6,r5

cmp r8,r4

bls loopt1

bx lr ;return
```

## Step 2. Look for reused registers

```
;gpio.asm
GPIO OFFSET = $200000
                                r0,r1,r2,r7
SETUP LED:
;param r0=BASE
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4]
                   ;set GPIO18 to output
bx lr
FLASH:
;param r0=BASE
;param r1 = number of flashes
orr r0,GPIO OFFSET
mov r7,r1
loop$:
  mov r1,#1
 lsl r1,#18
  str r1,[r0,#28] ;turn LED on
  mov r2,$0F0000 ; not using r2 for anything else sd
no need to push/pop it
   push {lr}
    bl TIMER
    pop {lr}
  mov r1,#1
  lsl r1,#18
  str r1,[r0,#40] ;turn LED off
  mov r2,$0F0000
  push {lr}
    bl TIMER
    pop {lr}
sub r7, r7, #1
```

```
cmp r7,#0
bgt loop$ ;end of outer loop. Runs r7 times
bx lr
```

; timer2\_2Param.asm

```
Delay: ;this function has 2 parameters

TIMER_OFFSET=$3000

mov r3,r0 ;BASE - depends on Pi model

orr r3,TIMER_OFFSET

mov r4,r1 ;$80000 passed as a parameter

ldrd r6,r7,[r3,#4]

mov r5,r6

loopt1: ;label still has to be different from one

in _start

ldrd r6,r7,[r3,#4]

sub r8,r6,r5

cmp r8,r4

bls loopt1

bx lr ;return
```

r0,r1,r3,r4,r5,r6,r7,r8

# Step 2. Look for required constants, labels and fix

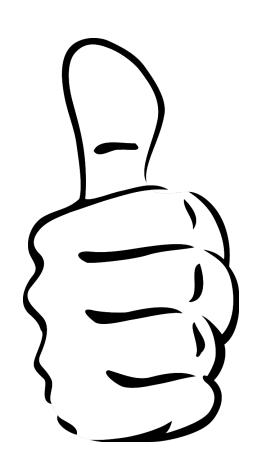
```
r0=BASE (1st
 ;gpio.asm
                                                      bl TIMER
                                                      pop {r0,r1,r7,lr}
GPIO OFFSET = $200000
                           param), but it's
SETUP LED:
                                                    sub r7, r7, #1
;param r0=BASE
                                                    cmp r7,#0
                          overwritten here
orr r0,GPIO OFFSET
                                                    bgt loop$ end of outer loop. Runs r7 times
mov r1,#1
                                                    bx lr
lsl r1,#24
str r1, [r0, #14]
                    ;set GPIO18 to output
                                                     ; timer2 2Param.asm
bx lr
FLASH:
                                                    Delay: ;this function has 2 parameters
                                                    TIMER OFFSET=$3000
;param r0=BASE
;param r1 = number of flashes
                                                    mov r3 r0 ;BASE
                                                                       - depends on Pi model
orr r0,GPIO OFFSET
                                                    orr r3,TIMER OFFSET
                                                    mov r4,r1 ;$80000 passed as a parameter
mov r7,r1
loop$:
                                                    ldrd r6,r7,[r3,#4]
  mov r1,#1
                                                    mov r5,r6
                                                    loopt1: ;label still has to be different from one
  lsl r1,#18
                                                    in start
  str r1,[r0,#28] ;turn LED on
                                                      ldrd r6, r7, [r3, #4]
   push {r0,r1,r7,lr} ;r0,r1,r7 in use push and then
                                                                           FLASH calls TIMER, but the
set parameters
                                                      sub r8, r6, r5
  mov r0,BASE
                                                      cmp r8,r4
                                                                           new timer is called Delay,
                                                      bls loopt1
  mov r1,$0F0000
                                                                           so we need to replace the
  bl TIMER <
                                                    bx lr ;return
  pop {r0,r1,r7,lr}
                                                                                label to branch to
  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

### Step 3. Test

```
r0=BASE (1st
 ;gpio.asm
GPIO OFFSET = $200000
                         param), backed up
SETUP LED:
;param r0=BASE
                                  here
orr r0,GPIO OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#4]
                   set GPIO18 to output
bx lr
FLASH:
;param r0=BASE
; need BASE for timer, so copy to r2 here
mov r2.r0
;param r1 = number of flashes
orr r0,GPIO OFFSET
mov r7,r1
loop$:
  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,$0F0000
  bl Delay
  pop {r0, 11, r7, lr}
sub r7, r7, #1
cmp r7,#0
bgt loop$
            end of outer loop. Runs r7 times
bx lr
 ; timer2_2Param.asm
Delay: ; this function has 2 parameters
TIMER OFFSET=$3000
mov r3, r0 ; BASE
                   - depends on Pi model
orr r3,TIMER OFFSET
mov r4,r1 \;$80000 passed as a parameter
ldrd r6, r7, [r3, #4]
mov r5,r6
loopt1: ;label still
                      FLASH calls TIMER, but the
in start
  ldrd r6, r7, [r3, \dag{4}]
                       new timer is called Delay,
  sub r8,r6,r5
                       so we need to replace the
  cmp r8,r4
  bls looptl
                            label to branch to
bx lr ;return
```





### Better software design

 Should put the includes in the files that use the function

- kernel7.img should include factorialj.asm and GPIO.asm
- GPIO.asm should include timer2\_2param.asm
- Should not have to rely on constants or registers being shared between files. (but they are, so keep labels and names unique).