How to separate out functions in Assembly code.

COS10004 CS Lab 9

1. 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"
```

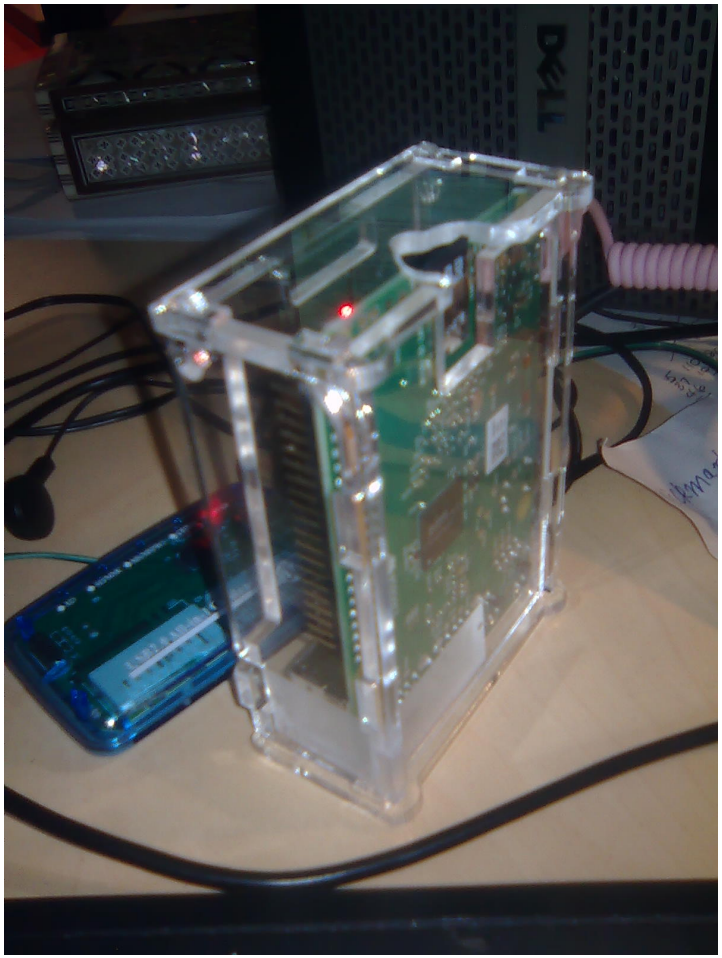
1. Open Kernel7.asm and separate out GPIO function for initialising LED

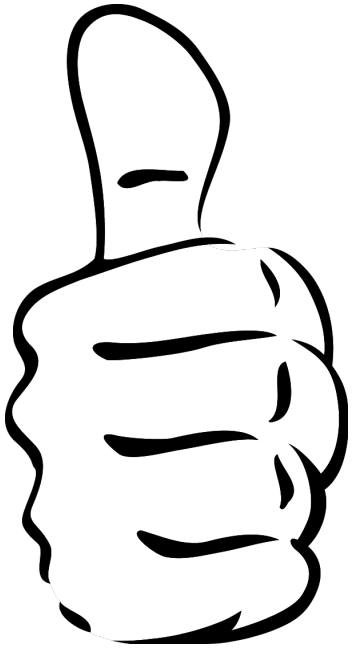
```
;Calculate
mov r1,#4 ;input
mov sp,$1000  ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer

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"
```

1. 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 fo 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
```

compile and test





2. 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
```

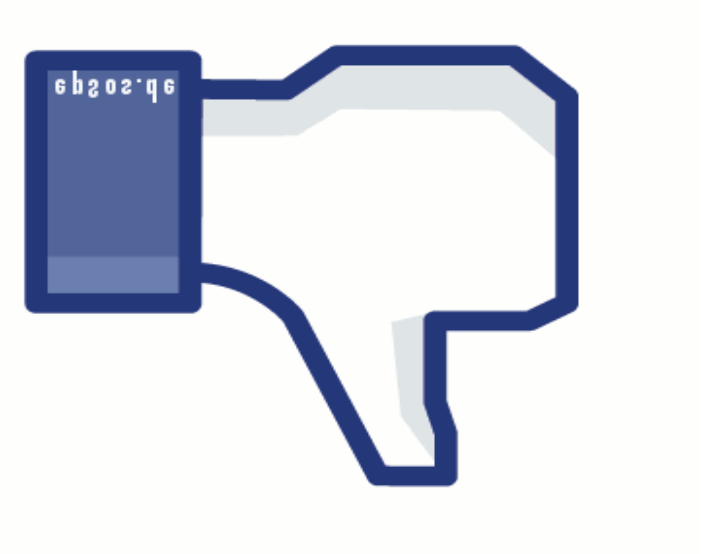
2. Separate out GPIO function for Flashing LED

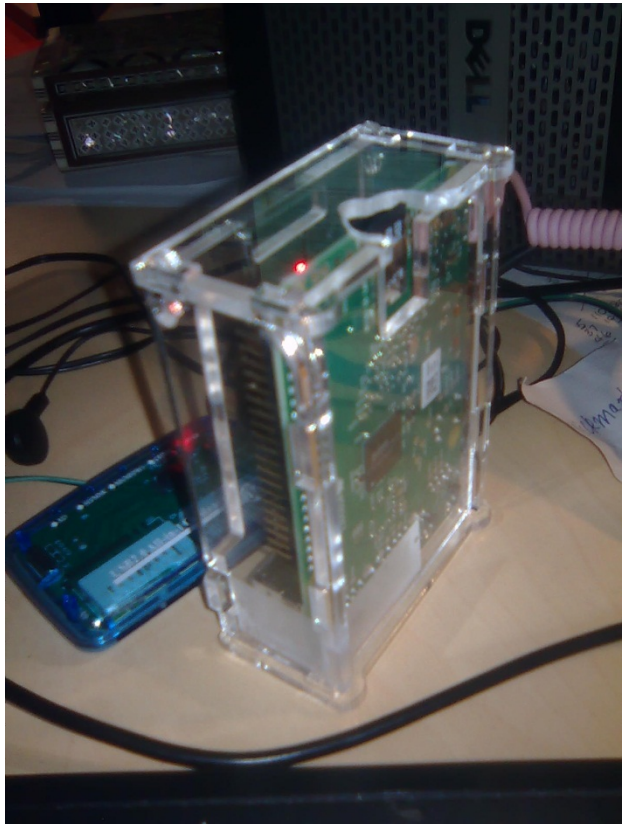
```
;Calculate
mov r1,#4 ;input
mov sp,$1000  ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer
BASE = $3F000000 ;RP2 ;GPIO_SETUP
GPIO_OFFSET = $200000
mov r0,BASE
bl SETUP_LED

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
  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

wait:
b wait
include "TIMER.asm"
include "factorialj.asm"
…
```

2. Separate out GPIO function for Flashing 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.

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
  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
```

compile and test





compile and test

- 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 lr 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 lr to pc)
   - wrong lr - correct one has been overwritten
10. return and run infinite loop

Solution

Edit the FLASH function to backup the value in "lr" 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}"
```

2. Separate out GPIO function for Flashing 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,$0F0000
     push {lr}
     bl TIMER
     pop {lr}
sub r7,#1
cmp r7,#0
bne  loop$   ;end of outer loop. Runs r7 times
bx lr
```

these fixed it.
ASM is weird.

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 lr
```

3. 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
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"
include "GPIO.asm"

;GPIO.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,$0F0000
   push {lr}
     bl TIMER
     pop {lr}
sub r7,r7,#1
cmp r7,#0
bgt  loop$  ;end of outer loop. Runs r7 times
```

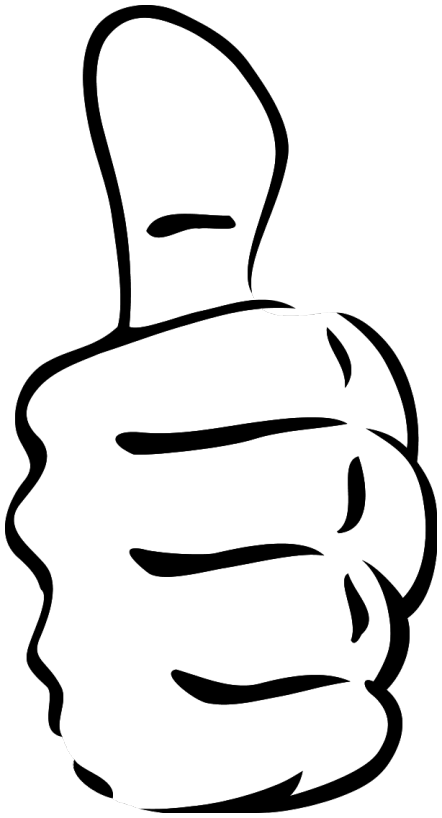
3. Move GPIO functions into their own file

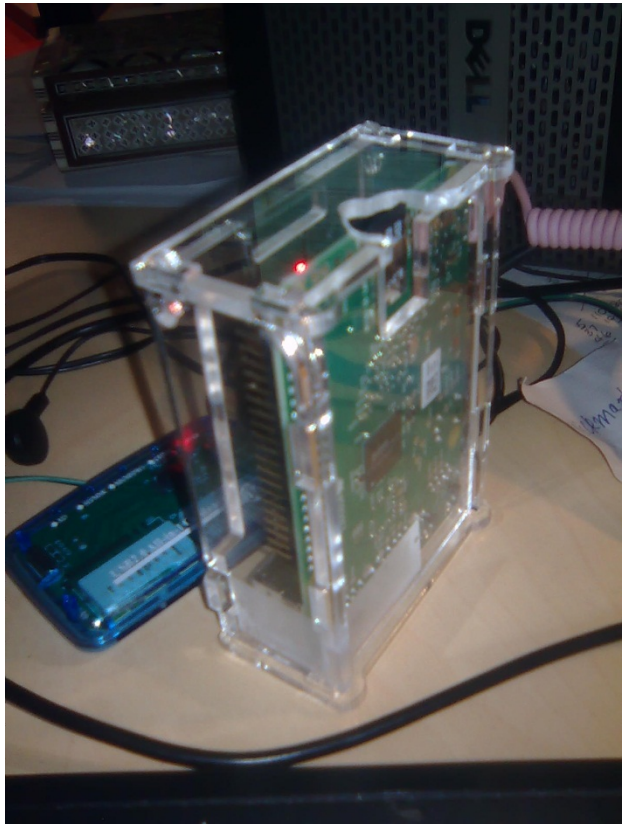
```
;kernel7.asm
;Calculate
mov r1,#4 ;input
mov sp,$1000  ;make room on the stack
mov r0,r1
bl FACTORIAL
mov r7,r0 ;store answer
BASE = $3F000000 ;RP2 ;GPIO_SETUP

mov r0,BASE
bl SETUP_LED

mov r0,BASE
bl SETUP_LED
mov r0,BASE
mov r1,r7
bl FLASH
wait:
b wait

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

;GPIO.asm
GPIO_OFFSET = $200000
```

We don't actually use GPIO_OFFSET in kernel7, so we'll move it to GPIO where it is relevant.

```
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
```

compile and test





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 = $3F000000;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
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 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
```

r0,r1,r2,r7

```
 ; 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

```
 ;gpio.asm
GPIO_OFFSET = $200000
SETUP_LED:
;param r0=BASE
orr r0,GPIO_OFFSET
mov r1,#1
lsl r1,#24
str r1,[r0,#14]      ;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
   push {r0,r1,r7,lr} ;r0,r1,r7 in use push and then
set parameters
  mov r0,BASE
  mov r1,$0F0000
  bl TIMER
  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,$0F0000
  bl TIMER
  pop {r0,r1,r7,lr}
sub r7,r7,#1
cmp r7,#0
bgt  loop$   ;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 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=BASE (1st param), but it's overwritten here

FLASH calls TIMER, but the new timer is called Delay, so we need to replace the label to branch to

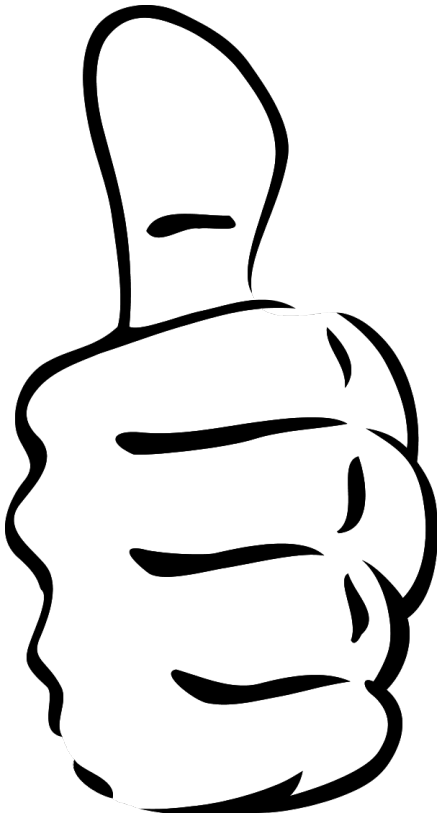
Step 3. Test

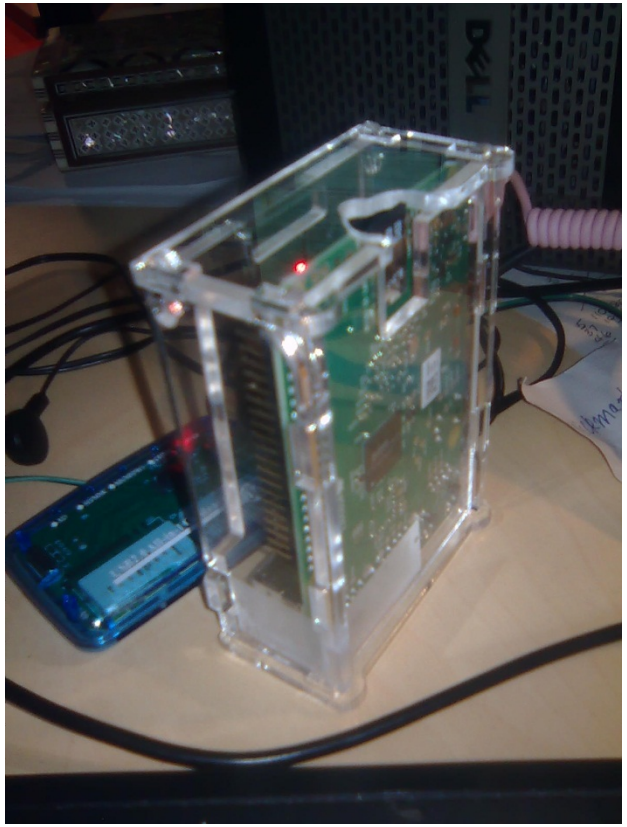
r0=BASE (1st param), backed up here

```
 ;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


 ; 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
in _start
  ldrd r6,r7,[r3,#4]
  sub r8,r6,r5
  cmp  r8,r4
  bls loopt1
bx lr   ;return
```

FLASH calls TIMER, but the new timer is called Delay, so we need to replace the label to branch to

compile and test





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).