COS10004 – Computer Systems

Name: Nguyen Manh Dung

Student ID: 104181789 (SWH01226)

LAB 10

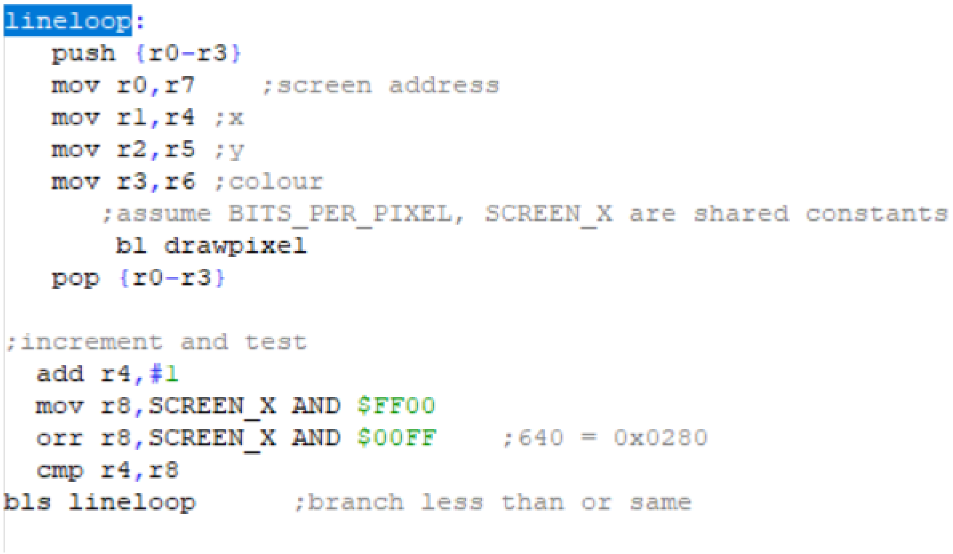
4. **Identify the parameters required when calling drawPixel:**

To draw a pixel, we need parameters r0-r3:

- The screen address is r0.
- The positions of r1 and r2 are x and y, respectively.
- The color of the pixel is r3.

5. Identify the code which controls the calling of drawPixel:

To draw a line its use the lineloop loop, which controls the call to drawpixel:

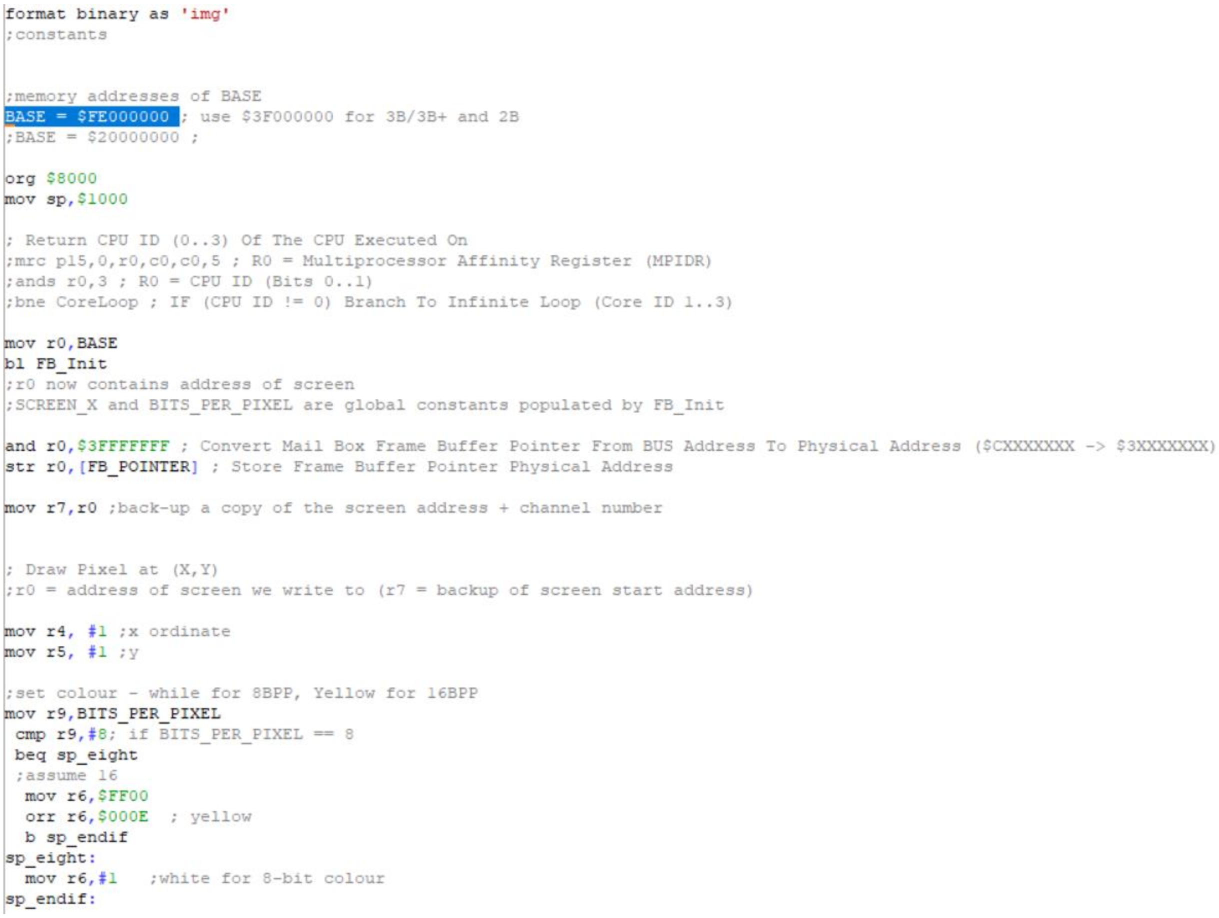


```
lineloop:
   push {r0-r3}
   mov r0,r7     ;screen address
   mov r1,r4 ;x
   mov r2,r5 ;y
   mov r3,r6 ;colour
      ;assume BITS_PER_PIXEL, SCREEN_X are shared constants
       bl drawpixel
   pop {r0-r3}

;increment and test
  add r4,#1
  mov r8,SCREEN_X AND $FF00
  orr r8,SCREEN_X AND $00FF    ;640 = 0x0280
  cmp r4,r8
bls lineloop      ;branch less than or same
```

In details, r4 and r5 respectively stands for the x and y position of the pixel. r0 is the screen address and r3 is the pixel colour. In this loop, r0, r5, and r3 remain constant. After a loop, r4 is increased in order to draw a pixel on a line at y position r5. After drawing a pixel in location (r4, r5), we proceed to draw further pixels in place (r4+1, r5) until r4 greater than r8, which is the screen's x limit.

6.



```
format binary as 'img'
;constants


;memory addresses of BASE
BASE = $FE000000 ; use $3F000000 for 3B/3B+ and 2B
;BASE = $20000000 ;

org $8000
mov sp,$1000

; Return CPU ID (0..3) Of The CPU Executed On
;mrc p15,0,r0,c0,c0,5 ; R0 = Multiprocessor Affinity Register (MPIDR)
;ands r0,3 ; R0 = CPU ID (Bits 0..1)
;bne CoreLoop ; IF (CPU ID != 0) Branch To Infinite Loop (Core ID 1..3)

mov r0,BASE
bl FB_Init
;r0 now contains address of screen
;SCREEN_X and BITS_PER_PIXEL are global constants populated by FB_Init

and r0,$3FFFFFFF ; Convert Mail Box Frame Buffer Pointer From BUS Address To Physical Address ($CXXXXXXX -> $3XXXXXXX)
str r0,[FB_POINTER] ; Store Frame Buffer Pointer Physical Address

mov r7,r0 ;back-up a copy of the screen address + channel number


; Draw Pixel at (X,Y)
;r0 = address of screen we write to (r7 = backup of screen start address)

mov r4, #1 ;x ordinate
mov r5, #1 ;y

;set colour - while for 8BPP, Yellow for 16BPP
mov r9,BITS_PER_PIXEL
 cmp r9,#8; if BITS_PER_PIXEL == 8
 beq sp_eight
 ;assume 16
  mov r6,$FF00
  orr r6,$000E  ; yellow
  b sp_endif
sp_eight:
  mov r6,#1   ;white for 8-bit colour
sp_endif:
```

kernel7.asm code of my shape drawer:

; Raspberry Pi B+,2 'Bare Metal' 16BPP Draw Pixel at any XY:

; 1. Setup Frame Buffer

;   assemble struct with screen requirements

;   receive pointer to screen or NULL

; 2. Start loop

;   Send pixel colour to location on screen

;   increment counter and loop if < 640

;note: r6 (colour) is 32-bit/4 byte register.

;at 16 bits/pixel, writing 32bits to adjacent pixels overwrites every second pixel.

```
; soln: write lower 2 bytes only (STRH) or lower byte(STRB).
;r0 = pointer + x * BITS_PER_PIXEL/8 + y * SCREEN_X * BITS_PER_PIXEL/8
format binary as 'img'
;constants


;memory addresses of BASE
BASE = $3F000000 ; use $3F000000 for 3B/3B+ and 2B
;BASE = $20000000 ;

org $8000
mov sp,$1000

; Return CPU ID (0..3) Of The CPU Executed On
;mrc p15,0,r0,c0,c0,5 ; R0 = Multiprocessor Affinity Register (MPIDR)
;ands r0,3 ; R0 = CPU ID (Bits 0..1)
;bne CoreLoop ; IF (CPU ID != 0) Branch To Infinite Loop (Core ID 1..3)

mov r0,BASE; r0 = BASE
bl FB_Init
;r0 now contains address of screen
;SCREEN_X and BITS_PER_PIXEL are global constants populated by FB_Init
```

```
and r0,$3FFFFFFF ; Convert Mail Box Frame Buffer Pointer From BUS Address To Physical Address ($CXXXXXXX -> $3XXXXXXX)

str r0,[FB_POINTER] ; Store Frame Buffer Pointer Physical Address



mov r7,r0 ;back-up a copy of the screen address + channel number




; Draw Pixel at (X,Y)

;r0 = address of screen we write to (r7 = backup of screen start address)



mov r6,#1   ;white for 8-bit colour



mov r4, #10

mov r5, #10



; Draw Box has top left = (10, 10) and bottom right = (18, 26)

draw_horizontal_line:

  push {r0-r3}

  mov r0,r7    ;screen address

  mov r1,r4 ;x

  mov r2,r5 ;y

  mov r3,r6 ;colour

     bl drawpixel

  pop {r0-r3}
 ;increment and test
 add r4,#1
 mov r8, #28
 cmp r4,r8
bls draw_horizontal_line      ;branch less than or same

draw_vertical_line:
  push {r0-r3}
  mov r0,r7    ;screen address
  mov r1,r4 ;x
  mov r2,r5 ;y
  mov r3,r6 ;colour
     bl drawpixel
  pop {r0-r3}

 ;increment and test
 add r5,#1
 mov r9, #37
 cmp r5,r9
bls draw_vertical_line      ;branch less than or same

mov r4, #10
mov r5, #10



draw_vertical_line2:
  push {r0-r3}
  mov r0,r7    ;screen address
  mov r1,r4 ;x
  mov r2,r5 ;y
  mov r3,r6 ;colour
     bl drawpixel
  pop {r0-r3}

 ;increment and test
 add r5,#1
 mov r8, #28
 mov r9, #36
 cmp r5,r9
bls draw_vertical_line2      ;branch less than or same

draw_horizontal_line2:
  push {r0-r3}
  mov r0,r7    ;screen address
  mov r1,r4 ;x
  mov r2,r5 ;y
  mov r3,r6 ;colour
     bl drawpixel
  pop {r0-r3}

 ;increment and test
 add r4,#1
 mov r8, #28
 cmp r4,r8
bls draw_horizontal_line2      ;branch less than or same


mov r4, #40
mov r5, #40

draw_vertical_line3:
  push {r0-r3}
  mov r0,r7    ;screen address
  mov r1,r4 ;x
  mov r2,r5 ;y
  mov r3,r6 ;colour
     bl drawpixel
  pop {r0-r3}


 ;increment and test
 add r5,#1
 mov r9, #80
 cmp r5,r9
bls draw_vertical_line3      ;branch less than or same



draw_horizontal_line3:
  push {r0-r3}
  mov r0,r7    ;screen address
  mov r1,r4 ;x
  mov r2,r5 ;y
  mov r3,r6 ;colour
     bl drawpixel
  pop {r0-r3}


 ;increment and test
 add r4,#1
 mov r8, #81
 cmp r4,r8
bls draw_horizontal_line3      ;branch less than or same
```

```
mov r4, #40

mov r5, #40


draw_diagonal_line:

   push {r0-r3}

   mov r0,r7    ;screen address

   mov r1,r4 ;x

   mov r2,r5 ;y

   mov r3,r6 ;colour

       bl drawpixel

   pop {r0-r3}


  ;increment and test

  add r4,#1

  add r5,#1

  mov r8, #80

  cmp r4,r8

bls draw_diagonal_line      ;branch less than or same
```

```
Loop:

  b Loop  ;wait forever


CoreLoop: ; Infinite Loop For Core 1..3

  b CoreLoop


include "FBinit8.asm"

include "drawpixel.asm"
```