



PRÁCTICA 1 DE ARQUITECTURA DE COMPUTADORES:

Diseño de un BenchMark sintético en Ensamblador y Cálculo de MIPS.

2º Ingeniería Informática - UHU
Arquitectura de Computadores
Profesor: Juan Manuel Ponce Real
Curso: 2020/2021

Alumnos
Alba Márquez Rodríguez
Alberto Fernández Merchán

Programa Pinta_Pixel

1/2

```
01 .MODEL small
02
03 .DATA
04 color db 01h
05 contador dw 00h
06 fin_linea dw 00h
07                                     ;280h
07                                     ;320d -> 140h x 2
08                                     ;modo grafico = 200 filas x 320 colum
09 npant dw 00h
10 aux dw 00h
11
12 .CODE
13     MOV AX, seg color
14     MOV DS, AX
15     MOV fin_linea, 00h
16
17     MOV AL, 13h      ;int sw para establecer el modo de pantalla en modo grafico
18     MOV AH, 0
19     INT 10h
20
21     MOV AX, 0A000h   ; Elegimos la direccion de memoria en modo grafico
22     MOV ES, AX
23
24     MOV SI, 280h     ;no se pinta en las dos primeras lineas
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58     MOV AH, 4ch
59     INT 21h
60 END
```

9 instrucciones fuera del bucle

11 instrucciones fuera del bucle

2 instrucciones fuera del bucle

Programa Pinta_Pixel

2/2

npant

Contador que muestra en las primeras líneas un píxel por cada pantalla empezada

```
25 reset_color:
26     MOV     color, 01h
27     JMP     pinta_pixel
28 cambia_color:
29     ADD     color, 01h
30     MOV     SI, 280h
31     MOV     contador, 00h
32     MOV     aux, SI
33     MOV     SI, npant
34     MOV     ES:[SI], AL
35     MOV     SI, aux
36     ADD     npant, 2
37     CMP     color, 07h
38     JAE     reset_color
```

Reinicio

Porque la pantalla se ha llenado

Contador fin de línea

Permanece en el bucle mientras no se llegue al fin de línea

Salto de línea

Saltar una línea = ADD SI, 140h
Para que se distinga mejor saltamos dos líneas

Contador

Controla que se llene la pantalla entera, si se llena, cambia el color y se reinicia todo

Parte del código que se ejecuta cuando se salta una línea

Cuando se llena una pantalla

```
39 pinta_pixel:
40     MOV     AL, color
41     MOV     ES:[SI], AL
42     ADD     SI, 2
43     ADD     fin_linea, 02h
44     CMP     fin_linea, 13Fh
45     JBE     pinta_pixel
46
47     ;hacemos salto de línea:
48     ADD     SI, 280h
49     MOV     fin_linea, 00h
50
51     ;incremento contador para saber cuando termina
52     ADD     contador, 01h
53     CMP     contador, 13Fh
54     JAE     cambia_color
55
56     JMP     pinta_pixel
```

Programa Pinta_Pixel

2/2

```
25 reset_color:
26     MOV     color, 01h
27     JMP     pinta_pixel
28 cambia_color:
29     ADD     color, 01h
30     MOV     SI, 280h
31     MOV     contador, 00h
32     MOV     aux, SI
33     MOV     SI, npant
34     MOV     ES:[SI], AL
35     MOV     SI, aux
36     ADD     npant, 2
37     CMP     color, 07h
38     JAE     reset_color
```

Si fin de pantalla:

+ 9 instrucciones

Si reinicia color:

+9+3 instrucciones

```
39 pinta_pixel:
40     MOV     AL, color
41     MOV     ES:[SI], AL
42     ADD     SI, 2
43     ADD     fin_linea, 02h
44     CMP     fin_linea, 13Fh
45     JBE     pinta_pixel
46
47     ;hacemos salto de linea:
48     ADD     SI, 280h ;500h
49     MOV     fin_linea, 00h
50
51     ;incremento contador para saber cuando termina
52     ADD     contador, 01h
53     CMP     contador, 13Fh
54     JAE     cambia_color
55
56     JMP     pinta_pixel
```

Bucle principal

6 instrucciones

Si fin de línea:

+5 instrucciones

Si fin de pantalla:

+5+1 instrucciones

Programa Residente: R.S.I.

```
01 .MODEL SMALL
02 .CODE
03 org 100h
04 Programa_Int:
05     JMP Reside
06
07     contador_int db 0
08
09 Rutina_Servicio PROC
10     CLI
11     INC contador_int
12     CMP contador_int, 91
13     JNE fin
14     MOV AH, 4ch
15     INT 21h
16 fin:
17
18     STI
19     IRET
20 ENDP
21
22 Reside:
23
24     MOV DX, OFFSET Rutina_Servicio
25     MOV AX, 0
26     MOV ES, AX ; ES vale 0.
27     MOV SI, 1Ch*4
28     CLI ; BIESTABLE QUE RESETEA LAS INTERRUPTONES.
29           ; deshabilita las interrupciones (clear interruption)
30     mov ES:[SI], DX ; 0000h:[1Ch*4] <- DX = offset rutina_tiempo
31     MOV ES:[SI+2], CS ; en la siguiente posicion de memoria tenemos el
32           ; segmento de codigo (CS) de la rutina tiempo
33     STI ; BIESTABLE QUE REESTABLECE LAS INTERRUPTONES
34           ; habilita las interrupciones (set interruption)
35     MOV DX, OFFSET Reside
36     INT 27h ; deja residente desde la etiqueta hacia atras (rutina)
37
38 END Programa_Int
```

Contador de Interrupciones.
Sale de la RSI cuando llega a 91 interrupciones.

18,2 interrupciones / 1 segundo --> 91 interrupciones / 5 segundos

6 instrucciones · 90 veces + 6 instrucciones · 1 vez = 546 instrucciones en 5 segundos

10 instrucciones

Ejecución del Código

E:\>

Número de pantallas que
ha completado:
 $40 * 15 = 600$ pantallas en
5 segundos

Cálculo de MIPS

127.004

Porque en la última pantalla no hace cambio de color

$$11 + (((6 \text{ inst} * 320 \text{ columnas}) - 1 + 5) * 66 + 9 - 1) * n^{\circ} \text{ pantallas} - 9 + 556$$

Fuera del bucle

Pintar una línea

Por cada pantalla

600

RSI

Programa pinta_pixel

RSI

Cada 7 pantallas -> + 3 instrucciones
 $600/7 = 85 * 3 = 255$

+255

Cálculo de MIPS

76.203.202 instrucciones en 5 segundos



15240640.4 instrucciones / segundo



15,24 MIPS