Nombre: Eduard Lorda Grupo: S2

Nombre: Miguel Gutiérrez

Hoja de respuesta al Estudio Previo

1. Explicad para qué sirven y qué operandos admiten las instrucciones:

psubb

Resta los enteros de byte empaquetados en mm/m64 de los enteros de byte empaquetados en mm.

pcmpgtb

Comparación con signo ≥ a nivel de Byte.

Ej: source 1 ≥ source 2 ⇒ dest

movdqa

Mueve un quadword double de op1 a op2, sin alineando a double.

Ej: Op1 = source Op2 = destino

movdqu

Mueve un quadword double de Op1 a Op2, sin alinear.

Ej: Op1 = source Op2 = destino

emms

Vacía el estado de MMX. Sirve para limpiar el estado de la FPU una vez acabada de usar.

2. La propiedad `__attribute__` y el atributo `aligned` sirven para:

__attribute__ permite especificar atributos especiales al hacer una declaración. Uno de ellos es aligned, especifica a cuantos Byte debe estar alineada la variable.

3. Programad en ensamblador una versión de la rutina que hay en `Procesar.c` procurando hacerla lo más rápida posible.

```
movl $0, %esi
movl 8(%ebp), %eax
movl 12(%ebp) %ebx
movl 16(%ebp) %ecx
movl 20 (%ebp) %edi
imull %edi, %edi
for:  cmpl %edi, %esi
      jge endfor
      movl (%ebx), %edx
      jubl (%eax), %edx
      movl %edx , (%ecx)
      cmpl $0, %edx
      jle else
if:   movl $255, (%ecx)
      jmp end-if
else: movl $0, (%ecx)
endif addl $16, %esi
      addl $16, %eax
```

```
addl $16, %ebx
addl $16, %ecx
jmp for
```

4. Explicad como se puede cargar un valor inmediato en un registro xmm usando la instrucción movdqu.

Cargo el inmediato a Memoria (16B) y después de Memoria (con la dirección del primer Byte) al xmm.

```
movdqu xmm0, [@base]
```

5. Programad en ensamblador una versión SIMD de la rutina que hay en `Procesar.c`.

```
movl $0, %esi
movl 8(%ebp), %eax
movl 12(%ebp), %ebx
movl 16(%ebp), %ecx
movl 20 (%ebp) %edi
imull %edi, %edi
for:  cmpl %edi, %esi
      jge end for
      movdqu (%eax), %xmm0
      movdqu (%ebx), %xmm1
      psubb %xmm1, %xmm0
      movdqu %xmm0, (%ecx)
      movdqu $zero, %xmm2
      pcmpgtb %xmm1, %xmm0
      movdqu %xmm0, (%ecx)
      addl $16, %esi
      addl $16, %eax
      addl $16, %ebx
      addl $16, %ecx
      jmp for
```

6. Escribid un código en ensamblador que a partir de un valor almacenado en un registro averigüe si es multiplo de 16.

```
movl $0x0F, %eax
andl %ecx, %eax
# valor en %ecx
# si eax = 0 => multiplo de 16
cmpl $0, %eax
jne no-multiplo
multiplo:
```