Nombre: Longbo Qui Grupo: 23 (4)

Nombre: Miguel Gutiérrez

Hoja de respuesta al Estudio Previo

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

pand

Ejecuta función de una AND bit a bit entre dos datos de 128 bits
Operand Origen AND Operand destino ⇒ Operand destino

pcmpgtb

Comparación con signo > a nivel de Byte
Ej: source1 > source2 ⇒ dest

movdqa

mueve un quadword double de Op1 a Op2, 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, %edi
movl $8(%ebp), %ebx
movl $12(%ebp), %ecx
movl $16(%ebp), %edx
imul %edx, %edx

for: cmpl %edx, %edi
     jge end_for
     movb (%ebx, %edi), %al
     andb $1, %al
     cmpb $0, %al
     jle else_1
     movb $255, (%ecx, %edi)
     jmp fi
else_1: movb $0, (%ecx, %edi)
fi: incl %edi
    jmp for
end_for:
```

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

Cargo el inmediato a Memoria (16 Bytes) 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`.

```
.data
uno: 1,1,...,1   (16)
cero: 0,0,...,0  (16)

movl $0, %edi
movl 8(%ebp), %ebx
movl 12(%ebp), %ebx
movl 16(%ebp), %edx
movl 20(%ebp), %esi
for: cmpl %edx, %edi
     jge end_for
for_1: cmpl %edx, %esi
     jge fi_for
     movdqu uno, %xmm0
     pand (%ebx, %edi), %xmm0
     pcmpgtb cero, %xmm0
     movdqu %xmm0, (%ecx, %esi)
     addl $16, %esi
     jmp for_1
fi_for: addl $16, %edi
        jmp for
end_for:
```

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

```
movl $0xF, %eax
andl %ecx, %eax
#valor en %ecx
#si eax = 0 → multiplo de 16
```