Nombre: Tanvir Hossain Grupo: 23

Nombre: Edgar Pérez Blanco

Hoja de respuesta al Estudio Previo

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

`pand`

AND bit a bit entre dos datos de 128 bit.
operando origen AND operando destino → operando destino

`pcmpgtb`

comparacion con signo mayor igual a nivel de Byte.

$source_1 \geq source_2 \rightarrow dest$

`movdqa`

move quad-word double de op1 a op2 (alineado a double)

Op_1 = source Op_2 = dest.

`movdqu`

move quad-word double de op1 a op2 (sin alinear)

Op_1 = source Op_2 = dest.

`emms`

empty mmx technology state (sin operandos)
Hace un clear de los tags (puestado a 1) de la x87FPU. Sirve para limpiar el estado de la FPU una vez acabada de usar.

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

__attribute__ sirve para añadir información a las declaraciones que para que el compilador las trate de una manera concreta.

aligned especifica una alineación concreta para dicha declaración.

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   # mat a
        movl  12(%ebp), %ebx  # mat b
        movl  16(%ebp), %ecx
        imul  %ecx, %ecx      # ecx = m^2
for:    cmpl  %ecx, %esi
        jge   endfor
        movb  (%eax, %esi), %dl
        andb  $1, %dl
        incl  %esi
```

```
if:     cmpb  $0, %dl
        jle   else
then:   movb  $255, -1(%ebx, %esi)
        jmp   for
else:   movb  $0, -1(%ebx, %esi)
end:    jmp   for
endfor:
```

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

cargando el inmediato a memoria (los 16 bytes) y despues de memoria (con la @ del primer byte) a la xmm.

movdqu xmm0, [@base]

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

```
        .data
unos:   .byte 1,1,1,1, 1,1,1,1, 1,1,1,1, 1,1,1,1
ceros:  .byte 0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0
        [...]
        movl  $0, %esi
        movl  8(%ebp), %eax   # mat a
        movl  12(%ebp), %ebx  # mat b
        movl  16(%ebp), %ecx
        imul  %ecx, %ecx      # ecx = m^2
for:    cmpl  %ecx, %esi
        jge   enfor
```

```
        movdqu  (unos), %xmm0
        pand    (%eax, %esi), %xmm0
        pcmpgtb (ceros), %xmm0
        movdqu  %xmm0, (%ebx, %edi)
        addl    $16, %esi
        jmp     for
endfor:
```

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

valor → %eax

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

```
andl  $0xF, %eax   # sin conservar el valor en el registro
```