Nombre: Daniel Martínez Bordes Grupo: 21

Nombre:

Hoja de respuesta al Estudio Previo

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

paddb

Suma los (bytes) empaquetados en el registro SRC origen con el destino la suma se hace en grupos de 8bits (Ori[0..7]+Dest[0..7]. Los registros son %.xmmi (128bits) o %.mmi (64bits) puede ser memoria también

movdqa

Mueve elementos del operando origen al destino (128bits) los operandos pueden ser %.xmmi o dirección de memoria alineada a 16.

movdqu

Mueve elementos del operando origen al destino (double quadword) pueden ser registros %.xmmi o posiciones de memoria (pueden no estar alineadas a 16 y no causar excepción

emms

Pone todos los valores de los registros MMX a vacío (todo 1) ya que estos registros estan mapeados (compartidos) con los FPU (floating-point)

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

- __atribute__ sirve para adjuntar caracteristicas a la declaración de funciones para hacer un check más extenso de errores p.e. permite asignar printf o scanf caracteristicas a la función para controlar parametros string en el codigo ...
- atributo aligned:

  variable __attribute__(aligned(multiple))

  Permite especificar la alineación mínima en memoria (en bytes) de la variable

pushl %ebp
movl %esp, %ebp

pushl %ebx
pushl %esi

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

movl $0, %eax # iterador bucle	sarb $4, %dl
movl 16(%ebp), %ecx # max bucle	movb %dl, (%esi, %eax)
imull %ecx, %ecx	inc %eax
movl 12(%ebp), %esi #matB	jmp for
movl 8(%ebp) %ebx #matA	end: popl %esi
for: cmpl %ecx, %eax	popl %ebx
jge end	movl %ebp, %esp
movb (%ebx, %eax), %dl	popl %ebp
	ret

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

Pasamos los 8 bytes de cada word como un immediato y lo guardamos en la pila y cuando estan los 16 guardados hacemos movdqu de la direccion base de la pila y el registro xmm y luego vaciamos la pila

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

pushl %ebp	for: cmpl %ecx, %eax
movl %esp, %ebp	jge end
pushl %ebx	movdqu (%ebx, %eax), %xmm0
pushl %esi	psllq $4, %xmm0
movl $0, %eax	movdqu %xmm0, (%esi, %eax)
movl 16(%ebp), %ecx	addl $16, %eax
imull %ecx, %ecx	jmp for
movl 12(%ebp), %esi #matB	end: emms
movl 8(%ebp), %ebx #matA	popl %esi
	popl %ebx
	movl %ebp, %esp
	popl %ebp

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

test $0x0000000F, %eax ?	(el dir nom enca que està (concat))
je end	