Procesamiento SIMD

Organización del Computador II

16 de Septiembre de 2014

Breve repaso SIMD

- Instrucciones aritméticas/lógicas datos empaquetados
  - *enteros:* `PADDx`, `PADDUSx`, `PADDUSx`, `PSUBUSx`, `PSUBSx`, `PMULLx`, `PMULHx`, `PHADDx`, etc.
  - *floats/doubles:* `ADDPx`, `ADDSx`, `SUBPx`, `SUBSx`, `MULPx`, `MULSx`, `ANDPx`, `ORPx`, `MAXPx`, `MAXSx`, `MINPx`, `MINSx`, etc.
- Instrucciones de desplazamiento
  - *empaquetados:* `PSRLx`, `PSLLx`, `PSRAx`, etc.
  - *no empaquetados:* `PSRLDQ`, `PSLLDQ`, etc.
- Intrucciones de empaquetado/desempaquetado
  - *desempaquetado:* `PUNPCKHxy`, `PUNPCKLxy`, etc.
  - *empaquetado:* `PACKSSxy`, `PACKUSxy`, etc.
- Instrucciones de comparación datos empaquetados
  - *enteros:* `PCMPEQx`, `PCMPGTx`, etc.
  - *floats/doubles:* `CMPPD`, `CMPPS`, `CMPSD`, `CMPSS`, etc.
- Otros ejercicios para pensar cómo hacer procesamiento simultáneo...

Ejercicio de Parcial

Sea una matriz de enteros sin signo de 16 bits de tamaño $n \times m$; donde n y m son múltiplos de 8. Se desea construir una función que obtenga la sumatoria de todos los valores dentro de la matriz que sean múltiplo de 4. Esta sumatoria entra en 29 bits.

```
int losMultiplosDe4(  unsigned int n,
                      unsigned short *matriz,
                      unsigned int m)
```

1. Programar en ASM usando instrucciones SIMD el código de la función pedida.
2. Explicar por qué su solución respeta la condición impuesta sobre las dimensiones de la matriz.

Nota: Un número es múltiplo de 4 $\iff$ los dos últimos dígitos de su representación binaria son 0.

Ejercicio de Parcial

Sea una matriz de enteros sin signo de 16 bits de tamaño $n \times m$; donde n y m son múltiplos de 8. Se desea construir una función que obtenga la sumatoria de todos los valores dentro de la matriz que sean múltiplo de 4. Esta sumatoria entra en 29 bits.

```
int losMultiplosDe4
(uInt n, unsigned short *matriz, uInt m)
```

¿Cómo hacemos para no "empaquetarnos" al intentar resolver un ejercicio de parcial como éste?

Vamos a ver una sugerencia de preguntas a realizarse.

Ejercicio de Parcial

Podemos empezar pensando cómo sería saber cuáles ints de un registro *xmmi* son múltiplos de 4.

1. ¿Cómo sería la operación suponiendo que ya tenemos los datos? (4 ints en *xmm*1).

```
mascara: dw 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC


movdqu  xmm14, [mascara]    ; mascara multiplos 4


pand    xmm1, xmm14         ; AND con 0xFFFC, los distintos a original no son 4x
pcmpeqw xmm1, xmm0          ; comparo original, iguales en 0xFFFF
                            ;    distintos en 0x0000
pand    xmm1, xmm0          ; AND original mascara anterior
                            ;     los no 4x quedan en 0x0000
                            ;     quedan sólo los 4x
```

Ejercicio de Parcial

2. ¿Cuál es la cantidad total de datos a procesar?

- n y m son múltiplos de 8. Como las columnas son multiplo de 8, en el procesamiento simultáneo no me van a quedar elementos sueltos que tenga que reajustar índices para procesarlos.
- No me olvido que n y m son `unsigned int`.

Ejercicio de Parcial

3. ¿Cómo hago el ciclo para procesarlos? ¿Necesito hacer fila/columna, o puedo procesar directo como vector?

- $n \times m$ es múltiplo de 8 y también encaja perfecto en el ciclado simultáneo.
- Entonces, puedo procesar la matriz como un vector ciclando hasta $n \times m$.

Ejercicio de Parcial

4. ¿Cuál es el tamaño de los datos a procesar? ¿Cuántos datos entran en un registro SSE?

- La matriz contiene elementos que son `unsigned short`.
- En un registro SSE (`XMMi`) entran 8 de estos elementos.

Ejercicio de Parcial

5. ¿Cuántos elementos puedo procesar simultáneamente? ¿Importa el cálculo que tengo que hacer en esa cantidad?

- En principio, puedo procesar (sumar) 8 elementos que son los que entran en `XMMi`. Pero tengo que hacer una sumatoria de todos, que podría no entrar en un `unsigned short` (2 bytes).
- Entonces las sumas voy a tener que hacerlas en `unsigned int` (4 bytes). Voy a tener que desempaquetar los elementos para hacer esas sumas.
- ¿Cómo lo hago?

Ejercicio de Parcial

5. ¿Cuántos elementos puedo procesar simultáneamente? ¿Importa el cálculo que tengo que hacer en esa cantidad?

- ¿Cómo lo hago?

```
movdqu      xmm2, xmm1      ; replico datos W para no perderlos
pxor        xmm15, xmm15    ; ceros para unpack
punpcklwd   xmm1, xmm15     ; xmm1 partes bajas desempaquetadas a DW
punpckhwd   xmm2, xmm15     ; xmm2 partes altas desempaquetadas a DW
```

Ejercicio de Parcial

6. ¿Cómo hago ese cálculo para procesarlos simultáneamente?

- Me fijo cuáles son múltiplos de 4.
- Tengo que sumarlos simultáneamente en un acumulador.
- No voy a poder hacer la suma simultánemente de 8 elementos `unsigned int`. Pero puedo hacer 2 sumas simultáneas de 4 elementos `unsigned int` cada una.
- **Pero primero:** voy a tener que "filtrar" los que no son múltiplos de 4.
- ¿Cómo hago todo esto simultáneamente?

Ejercicio de Parcial

6. ¿Cómo hago ese cálculo para procesarlos simultáneamente?

- Segundo hago las 2 sumas simultáneas de 4 `unsigned int` desempaquetando antes.

```
pxor    xmm15, xmm15      ; ceros para unpack

movdqu xmm2, xmm1         ; replico para no perder datos

punpcklwd xmm1, xmm15     ; xmm1 partes bajas desempaquetadas a DW
punpckhwd xmm2, xmm15     ; xmm2 partes altas desempaquetadas a DW

paddd xmm12, xmm1         ; acumulo partes bajas
paddd xmm13, xmm2         ; acumulo partes altas
```

Ejercicio de Parcial

7. ¿Cómo tengo que devolver o almacenar el resultado?

- El tipo de datos a retornar es `Int`. Entonces se devuelve por `EAX`.
- Si fuera otro tipo de dato debería empaquetar/desempaquetar según corresponda.
- Si fuera otra estructura parecida a la original (`unsigned short`) tendría que volver a empaquetar para guardar.

Ejercicio de Parcial

8. ¿Voy a necesitar variables locales (*stack*)?

- En este caso no. Pero en algún otro caso podría ser, y no me olvido de tenerlo en cuenta.

Ejercicio de Parcial

¿Cómo hacemos para no "empaquetarnos" al intentar resolver un ejercicio de parcial como éste?

Sugerencia de preguntas a hacernos:

1. ¿Cuál es la cantidad total de datos a procesar?
2. ¿Cómo hago el ciclo para procesarlos? ¿Necesito hacer fila/columna, o puedo procesar directo como vector?
3. ¿Cuál es el tamaño de los datos a procesar? ¿Cuántos datos entran en un registro SSE?
4. ¿Cuántos elementos puedo procesar simultáneamente? ¿Importa el cálculo que tengo que hacer en es cantidad?
5. ¿Cómo hago ese cálculo para procesarlos simultáneamente?
6. ¿Cómo tengo que devolver o almacenar el resultado?
7. ¿Voy a necesitar variables locales (*stack*)?

Ejercicio de Parcial - Resolución 1)

Definimos las etiquetas y preparamos los registros para el ciclo:

```
mascara: dw 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC, 0xFFFC


losMultiplosDe4:
; edi = n, rsi = matriz, edx = m

    xor rcx, rcx
    mov ecx, edi

    xor rax, rax
    mov eax, edx

    mul rcx                         ; rax = n x m
    mov rcx, rax                    ; rcx = n x m
    shr rcx, 3                      ; rcx = (n x m)/8

    pxor   xmm15, xmm15             ; ceros para unpack
    movdqu xmm14, [mascara]         ; mascara multiplos 4
    pxor   xmm13, xmm13             ; acumulador alta en 0
    pxor   xmm12, xmm12             ; acumulador baja en 0
```

Ejercicio de Parcial - Resolución 1)

Hacemos el ciclo:

```
.ciclo:
    movdqu xmm0, [rsi]        ; traigo 8 elementos W
    movdqu xmm1, xmm0         ; replico para no perder datos

    pand xmm1, xmm14          ; and con 0xFFFC, los que son
                              ;   distintos del original no
                              ;   son multiplos de 4

    pcmpeqw xmm1, xmm0        ; comparo por igualdad con
                              ;   originales y me queda mascara
                              ;   de 0xFFFF en los que no cambiaron

    pand xmm1, xmm0           ; and con 0xFFFF en multiplos de 4
                              ;   y con 0x0000 en no multiplos.
                              ;   Me quedan sólo los multiplos

    movdqu xmm2, xmm1         ; replico para no perder datos
    punpcklwd xmm1, xmm15     ; xmm1 partes bajas desempaquetadas a DW
    punpckhwd xmm2, xmm15     ; xmm2 partes altas desempaquetadas a DW

    paddd xmm12, xmm1         ; acumulo partes bajas
    paddd xmm13, xmm2         ; acumulo partes altas

    add rsi, 16               ; acomodo puntero y ciclo
    loop .ciclo
```

Ejercicio de Parcial - Resolución 1)

Devolvemos resultado:

```
    phaddd xmm12, xmm13    ; acumulacion parcial
    phaddd xmm12, xmm12    ; acumulacion parcial
    phaddd xmm12, xmm12    ; acumulacion final en 32bits bajos de xmm12

    xor  rax, rax          ; retorno Int por eax
    movd eax, xmm12        ; eax = sumatoria = 32bits bajos de xmm12

    ret
```

Ejercicio de Parcial - Resolución 2)

2. Explicar por qué su solución respeta la condición impuesta sobre las dimensiones de la matriz.

- Puedo considerar a la matríz de $n \times m$ como un vector de tamaño $n \times m$. Como n y m son múltiplos de 8, $n \times m$ va a ser múltiplo de 8. Como proceso de a 8 elementos respeto las condiciones sobre las dimensiones de la matriz.