ST0254 – Organización de computadores
Práctica 3: Procesamiento paralelo de cadenas de caracteres

Anderson Cano Castaño
Mario Giraldo Restrepo
Mateo Restrepo Restrepo

Departamento de Informática y Sistemas
Universidad EAFIT
Medellín, Colombia, Suramérica

RESUMEN
El presente documento contiene de forma corta y simple, la solución dada a uno de los problemas típicos de ciencias de la computación sobre las cadenas de texto el cual consiste en convertir un texto a mayúsculas o minúsculas esta vez optimizando su proceso mediante instrucciones MMX.

ST0254 – Organización de computadores en el semestre 2014-1.
PALABRAS CLAVE
Assembler, strings, Intel, Linux, Assembler, Bash.

2. OBJETIVOS

1. OBJETIVO GENERAL

Realizar el proceso de construcción de un programa que realice el procesamiento y conversión de un texto de mayúsculas o minúsculas y viceversa mediante comandos MMX.

2. OBJETIVOS ESPECIFICOS

Crear un programa en assembler para Linux que realice las siguientes tareas:

1. Lea mediante línea de comandos, la instrucción a realizar. Por ejemplo:

```
practica < archivo.txt > nuevoarchivo.txt
```

Este comando convierte a minúscula todo el contenido del archivo "archivo.txt" y deja la respuesta en el archivo "nuevoarchivo.txt"

```
practica l < archivo.txt > nuevoarchivo.txt
```

Este comando convierte a minúscula todo el contenido del archivo "archivo.txt" y deja la respuesta en el archivo "nuevoarchivo.txt"

2. Abra el archivo especificado en la línea de comandos.

3. Lea el archivo y mediante instrucciones MMX convierta dicho archivo en mayúsculas o minúsculas según sea la instrucción dada.

4. Guarde en un archivo nuevo, la salida.

5. Si hay errores, estos deben mostrarse en la salida estándar de errores.

6. Validar el tiempo de ejecución de la nueva rutina comparada con el comando tr para un archivo de texto grande.

3. ESTADO DE LA PRÁCTICA

La práctica se encuentra concluida y terminada totalmente.

4. FUNCIONAMIENTO

_mayus:

```
mov ebx, 0
mov eax, 3
mov ecx, buffer
mov edx, 512
int 0x80
cmp     eax, 0
je      _exit
mov esi,eax
mov esp,0
```

_cicloMayus:

```
movups  xmm1,[dMin1]
movups  xmm2,[dMin2]
movups  xmm3,[d0]

mov eax,esp
mov ebp,8
mul ebp
movups  xmm0,[buffer+eax]
movups  xmm4,xmm0
pcmpgtb xmm2,xmm0
pcmpgtb xmm4,xmm1
pand    xmm4,xmm2
pand    xmm4,xmm3
psubb   xmm0,xmm4
movups  [buffer+eax],xmm0

cmp esp,64
jge _outMayus
inc esp
jmp _cicloMayus
```

_minus:

```
mov ebx, 0
mov eax, 3
mov ecx, buffer
mov edx, 512
int 0x80
cmp     eax, 0
je      _exit
mov esi,eax
mov esp,0
```

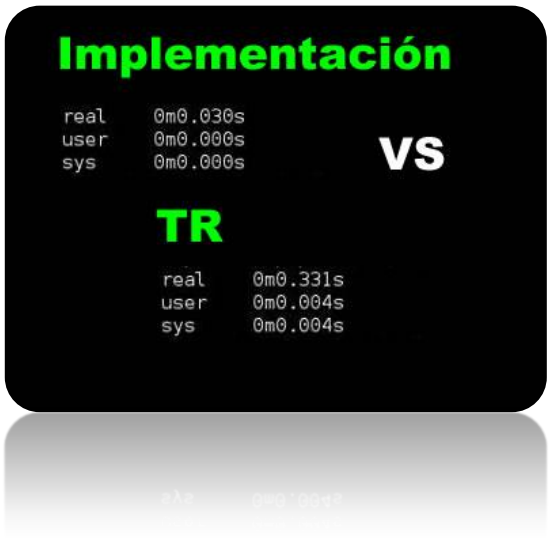
_cicloMinus:

```
movups  xmm1,[dMay1]
movups  xmm2,[dMay2]
movups  xmm3,[d0]
mov eax,esp
mov ebp,8
mul ebp
movups  xmm0,[buffer+eax]
movups  xmm4,xmm0
pcmpgtb xmm2,xmm0
pcmpgtb xmm4,xmm1
pand    xmm4,xmm2
pand    xmm4,xmm3
paddb   xmm0,xmm4
movups  [buffer+eax],xmm0
cmp esp,64
jge _outMinus
inc esp
jmp _cicloMinus
```

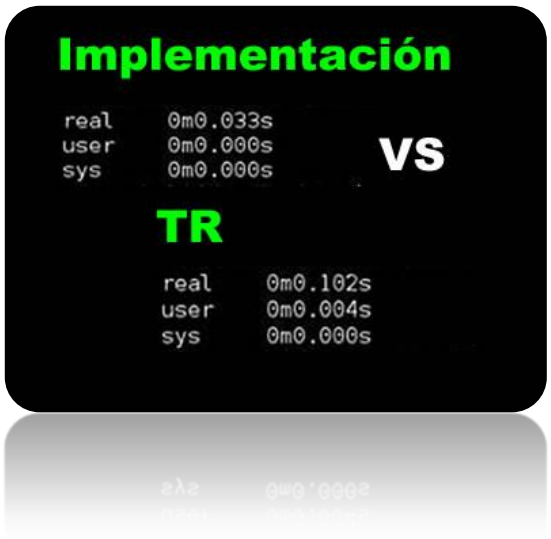
Tiempos De Ejecución

Para el test se utilizó un texto conformado con 4294 palabras.

Texto convertido a Minúsculas.



Texto convertido a Mayúsculas.



5. DIFICULTADES

- La inexperiencia en el manejo de instrucciones MMX.

- El cambio de paradigma aún es una dificultad presente, a pesar de que se ha venido trabajando durante el curso.

- Desconocimiento de optimizaciones para hacer más rápido nuestro programa.

6. POSIBLES MEJORAS

- Optimizar los tiempos de ejecución con respecto al tr.

- Mejor uso del espacio de los registros MMX.

7. REFERENCIAS

- [1] http://en.wikibooks.org/wiki/X86_Assembly/SSE
- [2] http://es.wikipedia.org/wiki/Netwide_Assembler
- [3] http://en.wikipedia.org/wiki/MMX_(instruction_set)
- [4]
- http://softpixel.com/~cwright/programming/simd/mmx.php
- [5] http://en.wikibooks.org/wiki/X86_Assembly/NASM_Syntax
- [6] http://www.nasm.us/