Índice de contenido

Proyecto Asembler
SERIE FIBONACCI
en GNU/Linux

1 Introducción

Como Tema de Trabajo final para el curso de Arquitectura de Computadores desarrollaremos un proyecto usando asembler y linux.

2 Problema

El problema trata de resolver la Serie de Fibonacci usando el lenguaje de Asembler usando el compilador nasm[1] en un GNU/Linux Debian de kernel 2.6.18-4-686.

3 Código Fuente

```
;
; Universidad Nacional de Ingenieria
; Arquitectura de Computadores
; Presentacion de la Serie de Fibonacci
;
;*****************************************************************************
;
; nasm -o linux.o -f elf linux.asm
; ld -s -o linux linux.o
;
;*****************************************************************************
; Constantes a Modificar
;*****************************************************************************
;
        maxTerms    equ 50 ; numero de terminos de la serie a calcular

;*****************************************************************************
;
; digitos = terminos * log(phi) + log(sqrt(5))/log(phi)
;
```

1- the Netwide Assembler - portable 80x86 assembler

```
; Numero de digitos que se usaran en el proceso de calculo
;
      digits        equ (maxTerms*209+1673)/1000
;
;
        cntDigits   equ 8       ; numero de digitos a contar

section .text
   global _start
;****************************************************************************
;****************************************************************************
_start:
; Iniciando dos numeros continuos
;
        mov     eax,'0000'              ; initializando todos en ceros ASCII
        mov     edi,num1                ;
        mov     ecx,1+digits/2          ; Para 4 bytes se tomara un tiempo mayor
        cld                     ; Iniciando de baja a alta memoria
        rep     stosd           ; escribiendo los datos
        inc     eax             ; haciendo ceros
        mov     [num2 + digits - 1],al ; el ultimo digito es uno
        mov     [counter + cntDigits - 1],al ;
        mov     dword [msg2],middle

        mov     ebp,[msd2]
        jmp     .bottom                 ; terminamos la inicializacion y comenzamos

.top
        ; agregando los numeros num2 a num1
        mov     edi,num1+digits-1
        mov     esi,num2+digits-1
        mov     ecx,digits      ;
        mov     ebp,[msd1]      ;
        call    AgregandoNumeros ; num1 += num2
        mov     [msd1],ebp      ;
        inc     ebp
        mov     ebx,num1+digits
        sub     ebx,ebp                 ; recogiendo el tamanhio en ebx
        call    Impresion       ;
        dec     dword [term]  ; decrementando el bucle
        jz      .done           ;

        ; agregando los numeros num1 a num2
        mov     edi,num2+digits-1
        mov     esi,num1+digits-1
        mov     ecx,digits      ;
        mov     ebp,[msd2]      ;
        call    AgregandoNumeros  ; num2 += num1
        mov     [msd2],ebp      ;
        inc     ebp
.bottom
        mov     ebx,num2+digits
        sub     ebx,ebp                 ; recogiendo el tamanhio en ebx
        call    Impresion       ;
        dec     dword [term]  ; decrementando el bucle
        jnz     .top            ;
.done
        call    CRLF            ; finalizando con CRLF
```

```
        mov     ebx,0           ; saliendo del cidig oexit
        mov     eax,1h          ; sys_exit
        int     80h             ;

;*****************************************************************************
;
; Impresion
;
; Fibonacci(1): 1
; Fibonacci(2): 1
; Fibonacci(3): 2
; Fibonacci(4): 3
;
; DESTRUIR: eax, ecx, edx, edi
;
;*****************************************************************************
Impresion:
        push    ebx
        mov     ecx,eol         ; imprimiendo y combinando CRLF y  msg1
        mov     edx,msg1len+eollen   ;
        call    ImpresionCadena      ;

        mov     edi,counter     ; imprimiendo el contador  and msg2
        mov     ecx,cntDigits + msg2len      ;
        call    ImpresionNumeroCadena

        call    IncrementandoContador      ; Tambien incrementando el contador

        mov     ecx,ebp                 ; obtiendo el puntero
        pop     edx             ; encontrando el tamanhio
        jmp     ImpresionCadena      ;

;*****************************************************************************
;
; ImprimesionNumeroCadena
;
; ENTRADA:    ds:edi ==> numero cadena, ecx = maximo tamanhio de cadena
; SALIDA:    CF , EAX = error en recoger el codigo
; DESTRUIR: eax, ecx, edx, edi
;
;*****************************************************************************
ImpresionNumeroCadena:
        ; Primero scanear el primer byte no cero
        mov     al,'0'          ;
        cld                     ; scaneando desde MSD to LSD
        repe    scasb           ;
        mov     edx,ecx                 ;
        mov     ecx,edi         ; colocando los byte despues
        dec     ecx             ; regresando
        inc     edx             ;

;*****************************************************************************
;
; ImpresionCadena
;
; ENTRADA:    ds:ecx ==> cadena, edx = tamanhio de cadena
; SALIDA:    CF , EAX = error
; DESTROYED: eax
```

```
;
;*************************************************************************
ImpresionCadena:
        mov     ebx, 1          ;
        mov     eax, 4h        ;
        int     80h             ; ignorando el valor de retorno
        ret                     ;

;*************************************************************************
;
; AgregandoNumeros
;
;
; ENTRADA:     es:edi ==> num1, ds:esi ==> num2, ecx= maximo tamanhio
;              ds:ebp ==> msd de num 1
; SALIDA:    CF set on overflow
; DESTRUIR: eax, esi, edi
;
;*************************************************************************
AgregandoNumeros:
        dec     ebp             ;
        std                     ; go from LSB to MSB
        clc                     ;
        pushf                   ; save carry flag
.top
        mov     eax,0f0fh       ; convert from ASCII BCD to BCD
        and     al,[esi] ; get next digit of number2 in al
        and     ah,[edi]        ; get next digit of number1 in ah
        popf                    ; recall carry flag
        adc     al,ah           ; add these digits
        aaa                     ; convert to BCD
        pushf                   ;
        add     al,'0'          ; convert back to ASCII BCD digit
        stosb                   ; save it and increment both counters
        dec     esi             ;
;
        cmp     edi,ebp                 ; are we at a new significant digit?
        loopnz .top             ; keep going until we've got them all
        cmp     al,'0'          ; is it a zero?
        jnz     .done           ; yes, so keep
        inc     ebp             ;
.done
        popf                    ; recall carry flag
        ret                     ;

;*************************************************************************
;
; IncrementandoContador
;
; ENTRADA:
; SALIDA:    CF set on overflow
; DESTRUIR: eax, ecx, edi
;
;*************************************************************************
IncrementandoContador:
        mov     ecx,cntDigits  ;
        mov     edi,counter+cntDigits-1
        std                     ; ir de  LSB a MSB
```

```
        stc                     ; esto es nuestro incremento
        pushf                   ; salvando
.top
        mov     ax,000fh        ; convirtiendo desde ASCII BCD a BCD
        and     al,[edi]; obteniendo los digitos de conteno
        popf                    ; llamando a carry
        adc     al,ah           ; agregando los digitos
        aaa                     ; convirtiendo a BCD
        pushf                   ;
        add     al,'0'          ; convirtiendo a ASCII BCD
        stosb                   ; guardando el incremento
        loop    .top            ;
        popf                    ; rellamando a carry
        ret                     ;

;*****************************************************************************
;
; CRLF
;
;
; ENTRADA:     ninguna
; SALIDA:    CF , AX
; DESTRUIR: ax, bx, cx, dx
;
;*****************************************************************************
CRLF:   mov     ecx,eol         ;
        mov     edx,eollen      ;
        jmp     ImpresionCadena      ;

;*****************************************************************************
; data
;*****************************************************************************
section .data
eol     db  10                  ; Lineas con Estilo de Linux
eollen  equ $ - eol

msg1   db  'Fibonacci('         ;
msg1len        equ $ - msg1
;*****************************************************************************
term dd maxTerms                ;
;*****************************************************************************
counter times cntDigits db '0'

middle  equ '): '
msg2   dd  0 ;en medio                  ;
msg2len        equ 3

msd1 dd num1+digits-1                   ; apuntando al digito mas significativo de num1
msd2 dd num2+digits-1                   ; apuntando al digito mas significativo de num2

num1 times digits db 0
num2 times digits db 0
overrun times 4 db 0            ; extrayendo el espacio
```

4 Resultado

Fibonacci(1): 1
Fibonacci(2): 1
Fibonacci(3): 2
Fibonacci(4): 3
Fibonacci(5): 5
Fibonacci(6): 8
Fibonacci(7): 13
Fibonacci(8): 21
Fibonacci(9): 34
Fibonacci(10): 55
Fibonacci(11): 89
Fibonacci(12): 144
Fibonacci(13): 233
Fibonacci(14): 377
Fibonacci(15): 610
Fibonacci(16): 987
Fibonacci(17): 1597
Fibonacci(18): 2584
Fibonacci(19): 4181
Fibonacci(20): 6765
Fibonacci(21): 10946
Fibonacci(22): 17711
Fibonacci(23): 28657
Fibonacci(24): 46368
Fibonacci(25): 75025
Fibonacci(26): 121393
Fibonacci(27): 196418
Fibonacci(28): 317811
Fibonacci(29): 514229
Fibonacci(30): 832040
Fibonacci(31): 1346269
Fibonacci(32): 2178309
Fibonacci(33): 3524578
Fibonacci(34): 5702887
Fibonacci(35): 9227465
Fibonacci(36): 14930352
Fibonacci(37): 24157817
Fibonacci(38): 39088169
Fibonacci(39): 63245986
Fibonacci(40): 102334155
Fibonacci(41): 165580141
Fibonacci(42): 267914296
Fibonacci(43): 433494437
Fibonacci(44): 701408733
Fibonacci(45): 1134903170
Fibonacci(46): 1836311903
Fibonacci(47): 2971215073
Fibonacci(48): 4807526976
Fibonacci(49): 7778742049
Fibonacci(50): 12586269025

5 Conclusiones

EL algoritmo desarrollador en asembler es altamente eficiente resolviendose para los 50 numeros en:

real 0m0.004s

user 0m0.000s

sys 0m0.000s