

Práctica 01

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
01	Introducción	2 días

1. Datos de los estudiantes

- Grupo: 2
- Integrantes:
 - Carnero Eduardo
 - Casaverde Diego
 - Laura Cesar
 - Zamata Jordy

2. Ejercicios

1. Redacta el siguiente código, genera el código ensamblador y explica en que parte (del código ensamblador) se definen las variables c y m.

Solución

```
.....
LC0:
    .ascii "abcdef\0" //aqui se define LC0
    .text
globl _main
    .def _main; .scl 2; .type 32; .endef
main:
    pushl %ebp
    movl %esp, %ebp
    andl $-16, %esp
    subl $16, %esp
    call main
    movl $LC0, 8(%esp) // aqui C obtiene el valor de LC0
    movl $11148, 12(%esp) // aqui m obtiene el el valor correspondiente
    movl %eax, %eax
    leave
    ret
```

2. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 8.

Solución

```
.....
Actividades Editor de textos
Abrir
1 .file "prueba.cpp"
2 .text
3 .section .rodata
4 .LC0:
5 .string "abcdef"
6 .text
7 .globl main
8 .type main, @function
9 main:
10 .LFB0:
11 .cfi_startproc
12 endbr64
13 pushq %rbp
14 .cfi_def_cfa_offset 16
15 .cfi_offset 6, -16
16 movq %rsp, %rbp
17 .cfi_def_cfa_register 6
18 leaq .LC0(%rip), %rax
19 movq %rax, -8(%rbp)
20 movl $11148, -16(%rbp)
21
22 movl -16(%rbp), %eax
23 leal 7(%rax), %edx
24 testl %eax, %eax
25 cmovs %edx, %eax
26 sarl $3, %eax
27 movl %eax, -12(%rbp)
28
29 movl $0, %eax
30 popq %rbp
31 .cfi_def_cfa 7, 8
32 ret
33 .cfi_endproc
34 .LFE0:
35 .size main, .-main
36 .ident "GCC: (Ubuntu 9.3.0-10ubuntu2) 9.3.0"
37 .section .note.GNU-stack,"",@progbits
38 .section .note.gnu.property,"a"
```

Se define la división entre 8

3. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 4.

Solución

```
.....
.section    __TEXT,__text,regular,pure_instructions
.build_version macos, 10, 15    sdk_version 10, 15, 6
.globl _main                    ## -- Begin function main
.p2align    4, 0x90

_main:                                ## @main
.cfi_startproc
## %bb.0:
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register %rbp
    xorl    %eax, %eax
    movl    $0, -4(%rbp)
    movl    %edi, -8(%rbp)
    movq    %rsi, -16(%rbp)
    movb    $97, -17(%rbp)
    movl    $11148, -24(%rbp)    ## imm = 0x2B8C
    movl    -24(%rbp), %ecx
    movl    %eax, -40(%rbp)    ## 4-byte Spill
    movl    %ecx, %eax
    cltd
    movl    $8, %ecx
    idivl    %ecx
    movl    %eax, -28(%rbp)
    movl    -24(%rbp), %eax
    cltd
    movl    $4, %ecx
    idivl    %ecx
    movl    %eax, -32(%rbp)
    movl    -24(%rbp), %eax
    cltd
    movl    $2, %ecx
    idivl    %ecx
    movl    %eax, -36(%rbp)
    movl    -40(%rbp), %eax    ## 4-byte Reload
    popq    %rbp
    retq
.cfi_endproc
                                ## -- End function

.subsections_via_symbols
```

División entre 4

4. Redacta el siguiente código, genera el código ensamblador y explica en qué parte (del código ensamblador) se define la división entre 2.

Solución

.....

```

movl $LC0, 28(%esp)
movl $11148, 24(%esp)
movl 24(%esp), %eax
leal 7(%eax), %edx
testl %eax, %eax
cmovs %edx, %eax
sarl $3, %eax

```

División entre 8

```

movl %eax, 20(%esp)
movl 24(%esp), %eax
leal 7(%eax), %edx
testl %eax, %eax
cmovs %edx, %eax
sarl $3, %eax

```

División entre 4

```

movl %eax, 16(%esp)
movl 24(%esp), %eax
movl %eax, %edx
shrl $31, %edx
addl %edx, %eax
sarl %eax

```

el shrl es encargado de dividir sin signo esto se debe a que es una división entre 2 por lo que lo mejor es quitarle un byte
División entre 2

5. Redacta el siguiente código, genera el código ensamblador y explica:

- En qué parte del código ensamblador se define la función div4.
- En qué parte del código ensamblador se invoca a la función div4.
- En qué parte del código ensamblador dentro de la función div4 se procesa la división.

Solución

```
Actividades Editor de textos
Abrir
*prueba.s
1 .file "prueba2.cpp"
2 .text
3 .globl _Z4div4i
4 .type _Z4div4i, @function
5 _Z4div4i:
6 .LFB0:
7 .cfi_startproc
8 endbr64
9 pushq %rbp
10 .cfi_def_cfa_offset 16
11 .cfi_offset 6, -16
12 movq %rsp, %rbp
13 .cfi_def_cfa_register 6
14 movl %edi, -4(%rbp)
15 movl -4(%rbp), %eax
16 leal 3(%rax), %edx
17 testl %eax, %eax
18 cmovs %edx, %eax
19 sarl $2, %eax
20 popq %rbp
21 .cfi_def_cfa 7, 8
22 ret
23 .cfi_endproc
24 .LFE0:
25 .size _Z4div4i, .-_Z4div4i
26 .section .rodata
27 .LC0:
28 .string "abcdef"
29 .text
30 .globl main
31 .type main, @function
32 main:
33 .LFB1:
```

Definición de la función
div4

```
32 main:
33 .LFB1:
34     .cfi_startproc
35     endbr64
36     pushq %rbp
37     .cfi_def_cfa_offset 16
38     .cfi_offset 6, -16
39     movq %rsp, %rbp
40     .cfi_def_cfa_register 6
41     subq $32, %rsp
42     leaq .LC0(%rip), %rax
43     movq %rax, -8(%rbp)
44     movl $11148, -28(%rbp)
45     movl -28(%rbp), %eax
46     leal 7(%rax), %edx
47     testl %eax, %eax
48     cmovs %edx, %eax
49     sarl $3, %eax
50     movl %eax, -24(%rbp)
51     movl -28(%rbp), %eax
52     leal 3(%rax), %edx
53     testl %eax, %eax
54     cmovs %edx, %eax
55     sarl $2, %eax
56     movl %eax, -20(%rbp)
57     movl -28(%rbp), %eax
58     movl %eax, %edx
59     shr $31, %edx
60     addl %edx, %eax
61     sarl %eax
62     movl %eax, -16(%rbp)
63     movl $5, %edi
64     call Z4div4i
65     movl %eax, -12(%rbp)
66     movl $0, %eax
67     leave
68     .cfi_def_cfa 7, 8
69     ret
70     .cfi_endproc
```

Se llama a la función
div4

Actividades Editor de textos

Abrir

*prueba.s

```
1 .file "prueba2.cpp"
2 .text
3 .globl _Z4div4i
4 .type _Z4div4i, @function
5 _Z4div4i:
6 .LFB0:
7 .cfi_startproc
8 endbr64
9 pushq %rbp
10 .cfi_def_cfa_offset 16
11 .cfi_offset 6, -16
12 movq %rsp, %rbp
13 .cfi_def_cfa_register 6
14 movl %edi, -4(%rbp)
15 movl -4(%rbp), %eax
16 leal 3(%rax), %edx
17 testl %eax, %eax
18 cmovs %edx, %eax
19 sarl $2, %eax
20 popq %rbp
21 .cfi_def_cfa 7, 8
22 ret
23 .cfi_endproc
24 .LFE0:
25 .size _Z4div4i, .-_Z4div4i
26 .section .rodata
27 .LC0:
28 .string "abcdef"
29 .text
30 .globl main
31 .type main, @function
32 main:
33 .LFB1:
34 .cfi_startproc
35 endbr64
36 pushq %rbp
37 .cfi_def_cfa_offset 16
38 .cfi_offset 6, -16
39 movq %rsp, %rbp
```

Se procesa la división

6. Redacta el siguiente código, genera el código ensamblador y explica:

- En qué parte del código ensamblador se define la función div.
- En qué parte del código ensamblador se invoca a la función div.
- En qué parte del código ensamblador dentro de la función div se procesa la división.

Solución

```

movl    %eax, -52(%rbp)
movl    -44(%rbp), %eax
movl    %eax, %ecx
shrl    $31, %ecx
addl    %ecx, %eax
sarl    %eax
movl    %eax, -56(%rbp)

Ltmp0:
movl    $5, %edi
movl    $4, %esi
callq   __Z4divsii

Ltmp1:
movl    %eax, -84(%rbp)    ## 4-byte Spill
jmp     LBB2_1
LBB2_1:
movl    -84(%rbp), %eax    ## 4-byte Reload
movl    %eax, -60(%rbp)
Ltmp2:
movl    $5, %edi
callq   __Z4div4i
Ltmp3:
movl    %eax, -88(%rbp)    ## 4-byte Spill
jmp     LBB2_2
LBB2_2:
    ## @__Z4divsii
__Z4divsii:
    ## @__Z4divsii
.cfi_startproc
## %bb.0:
pushq   %rbp
.cfi_def_cfa_offset 16
.cfi_offset %rbp, -16
movq    %rsp, %rbp
.cfi_def_cfa_register %rbp
movl    %edi, -4(%rbp)
movl    %esi, -8(%rbp)
movl    -4(%rbp), %eax
ltd     %eax, %eax
idivl   -8(%rbp)
popq    %rbp
retq
.cfi_endproc
    ## -- End function

```

Llamado de la función div

idivl línea para la división


```
.section __TEXT,__text,regular,pure_instructions
.build_version macos, 10, 15 sdk_version 10, 15, 6
.globl __Z4divsii ## -- Begin function _Z4divsii
.p2align 4, 0x00
_Z4divsii: ## @_Z4divsii
.cfi_startproc
## %bb.0:
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset %rbp, -16
movq %rsp, %rbp
.cfi_def_cfa_register %rbp
movl %edi, -4(%rbp)
movl %esi, -8(%rbp)
movl -4(%rbp), %eax
cltd
idivl -8(%rbp)
popq %rbp
retq
.cfi_endproc
## -- End function

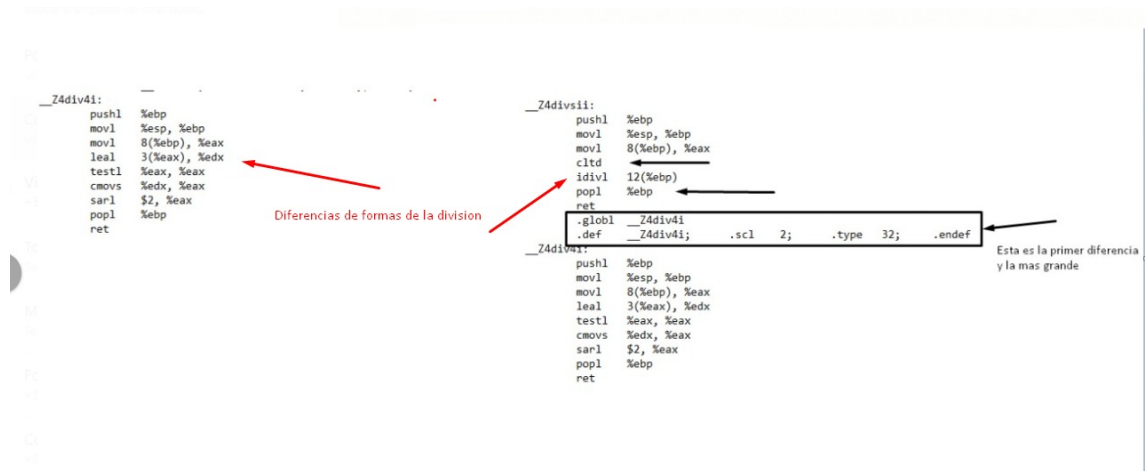
.globl __Z4div4i ## -- Begin function _Z4div4i
.p2align 4, 0x90
_Z4div4i: ## @_Z4div4i
.cfi_startproc
## %bb.0:
pushq %rbp
.cfi_def_cfa_offset 16
.cfi_offset %rbp, -16
movq %rsp, %rbp
.cfi_def_cfa_register %rbp
movl %edi, -4(%rbp)
movl -4(%rbp), %eax
cltd
movl $4, %ecx
idivl %ecx
popq %rbp
retq
.cfi_endproc
## -- End function
```

Definición de
la función
div

7. De las preguntas anteriores, se ha generado código por cada función, ambas dividen entre 4, pero difieren un poco en su implementación. Investigue a qué se debe dicha diferencia y comente cuáles podrían ser las consecuencias.

Solución

Las diferencias de código assembler usado se deben a la estructura de las funciones, la cual una divide por un numero estático y la otra es una división entre 2 variables enteras, las principales consecuencias son el espacio requerido y la cantidad de líneas por ende la velocidad de cada uno



The image shows a side-by-side comparison of two assembly functions, `__Z4div4i` and `__Z4divsi`, both performing division by 4. Red arrows point to specific differences in the code.

Left Function: `__Z4div4i`

```

__Z4div4i:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax
    leal    3(%eax), %edx
    testl   %eax, %eax
    cmovs   %edx, %eax
    sarl    $2, %eax
    popl    %ebp
    ret
    
```

Right Function: `__Z4divsi`

```

__Z4divsi:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax
    cld
    idivl   12(%ebp)
    popl    %ebp
    ret
    
```

Global Symbol Definition:

```

.globl __Z4div4i
.def __Z4div4i; .scl 2; .type 32; .endef
__Z4div4i:
    pushl   %ebp
    movl    %esp, %ebp
    movl    8(%ebp), %eax
    leal    3(%eax), %edx
    testl   %eax, %eax
    cmovs   %edx, %eax
    sarl    $2, %eax
    popl    %ebp
    ret
    
```

Annotations:

- A red arrow points from the text "Diferencias de formas de la division" to the `leal 3(%eax), %edx` instruction in `__Z4div4i` and the `idivl 12(%ebp)` instruction in `__Z4divsi`.
- A black box highlights the global symbol definition for `__Z4div4i` in the right column, with an arrow pointing to it from the text "Esta es la primera diferencia y la mas grande".