

## Práctica 01

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

PRÁCTICA	TEMA	DURACIÓN
01	Introducción	3 horas

### 1. Datos de los estudiantes

- Grupo: xxxxxx
- Integrantes:
  - Kevin Linares Salinas
  - xxxxxxxxxxxxxxxxx

### 2. Ejercicios

1. Redacta el siguiente código, genera con ensamblador y explica en qué parte (del código ensamblador) se definen las variables c y m. (2 puntos).

Solución:

```
int main(){  
    char* c = "abcdef";  
    int m = 11148;  
    return 0;  
}
```

```
1  .LC0:  
2      .string "abcdef"  
3  main:  
4      push    rbp  
5      mov     rbp, rsp  
6      mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0  
7      mov     DWORD PTR [rbp-12], 11148  
8      mov     eax, 0  
9      pop     rbp  
10     ret
```

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

Solución:

```
2  int main(){
3  char* c = "abcdef";
4  int m = 11148;
5  int x = m/8;
6  return 0;
7  }
```

```
1  .LC0:
2      .string "abcdef"
3  main:
4      push    rbp
5      mov     rbp, rsp
6      mov     QWORD PTR [rbp-8], OFFSET FLAT: .LC0
7      mov     DWORD PTR [rbp-12], 11148
8      mov     eax, DWORD PTR [rbp-12]
9      lea     edx, [rax+7]
10     test    eax, eax
11     cmovs   eax, edx
12     sar     eax, 3
13     mov     DWORD PTR [rbp-16], eax
14     mov     eax, 0
15     pop     rbp
16     ret
```

3. Redacta el siguiente código, genera el código ensamblador y explica en que parte (del código ensamblador) se define la división entre 7. (2 puntos). Solución:

```

1
2  int main(){
3  char* c = "abcdef";
4  int m = 11148;
5  int x = m/8;
6  int y = m/4;
7  int z = m/2;
8  return 0;
9  }

```

```

1  .LC0:
2      .string "abcdef"
3  main:
4      push    rbp
5      mov     rbp, rsp
6      mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0
7      mov     DWORD PTR [rbp-12], 11148
8      mov     eax, DWORD PTR [rbp-12]
9      lea     edx, [rax+7]
10     test    eax, eax
11     cmovs   eax, edx
12     sar     eax, 3
13     mov     DWORD PTR [rbp-16], eax
14     mov     eax, DWORD PTR [rbp-12]
15     lea     edx, [rax+3]
16     test    eax, eax
17     cmovs   eax, edx
18     sar     eax, 2
19     mov     DWORD PTR [rbp-20], eax
20     mov     eax, DWORD PTR [rbp-12]
21     mov     edx, eax
22     shr     edx, 31
23     add     eax, edx
24     sar     eax
25     mov     DWORD PTR [rbp-24], eax
26     mov     eax, 0
27     pop     rbp
28     ret

```

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

```

1  int main(){
2  char* c = "abcdef";
3  int m = 11148;
4  int x = m/8;
5  int y = m/4;
6  int z = m/2;
7  return 0;
8  }

```

```

1  .LC0:
2  .string "abcdef"
3  main:
4  push    rbp
5  mov     rbp, rsp
6  mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0
7  mov     DWORD PTR [rbp-12], 11148
8  mov     eax, DWORD PTR [rbp-12]
9  lea     edx, [rax+7]
10 test    eax, eax
11 cmovs   eax, edx
12 sar     eax, 3
13 mov     DWORD PTR [rbp-16], eax
14 mov     eax, DWORD PTR [rbp-12]
15 lea     edx, [rax+3]
16 test    eax, eax
17 cmovs   eax, edx
18 sar     eax, 2
19 mov     DWORD PTR [rbp-20], eax
20 mov     eax, DWORD PTR [rbp-12]
21 mov     edx, eax
22 shr     edx, 31
23 add     eax, edx
24 sar     eax
25 mov     DWORD PTR [rbp-24], eax
26 mov     eax, 0
27 pop     rbp
28 ret

```

5. Redacta el siguiente código, genera el código ensamblador y explica: (4 puntos):

<pre> 1  int div4(int x){ 2  return x/4; 3  } 4  int main(){ 5  char* c = "abcdef"; 6  int m = 11148; 7  int x = m/8; 8  int y = m/4; 9  int z = m/2; 10 int rpt = div4(5); 11 return 0; 12 } </pre>	<pre> 1  div4(int): 2      push    rbp 3      mov     rbp, rsp 4      mov     DWORD PTR [rbp-4], edi 5      mov     eax, DWORD PTR [rbp-4] 6      lea     edx, [rax+3] 7      test    eax, eax 8      cmovs   eax, edx 9      sar     eax, 2 10     pop     rbp 11     ret 12 13 .LC0: 14     .string "abcdef" 15 16 main: 17     push    rbp 18     mov     rbp, rsp 19     sub     rsp, 32 20     mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0 21     mov     DWORD PTR [rbp-12], 11148 22     mov     eax, DWORD PTR [rbp-12] 23     lea     edx, [rax+7] 24     test    eax, eax 25     cmovs   eax, edx 26     sar     eax, 3 27     mov     DWORD PTR [rbp-16], eax 28     mov     eax, DWORD PTR [rbp-12] 29     lea     edx, [rax+3] 30     test    eax, eax 31     cmovs   eax, edx 32     sar     eax, 2 33     mov     DWORD PTR [rbp-20], eax 34     mov     eax, DWORD PTR [rbp-12] 35     mov     edx, eax 36     shr     edx, 31 37     add     eax, edx 38     sar     eax 39     mov     DWORD PTR [rbp-24], eax 40     mov     edi, 5 41     call    div4(int) 42     mov     DWORD PTR [rbp-28], eax 43     mov     eax, 0 44     leave 45     ret </pre>
--	---

a) En que parte del código ensamblador se define la función div4.

```

push    rbp
mov     rbp, rsp
mov     DWORD PTR [rbp-4], edi

```

b) En que parte del código ensamblador se invoca a la función div4.

```

38     mov     edi, 5
39     call    div4(int)
40     mov     DWORD PTR [rbp-28], eax

```

c) En que parte del código ensamblador dentro de la función div4 se procesa la división.

```

5      mov     eax, DWORD PTR [rbp-4]
6      lea     edx, [rax+3]
7      test    eax, eax
8      cmovs   eax, edx
9      sar     eax, 2

```

6. Redacta el siguiente código, genera el código ensamblador y explica: (4 puntos):

```

1  div(int, int):
2      push    rbp
3      mov     rbp, rsp
4      mov     DWORD PTR [rbp-4], edi
5      mov     DWORD PTR [rbp-8], esi
6      mov     eax, DWORD PTR [rbp-4]
7      cdq
8      idiv    DWORD PTR [rbp-8]
9      pop     rbp
10     ret
11  div4(int):
12     push    rbp
13     mov     rbp, rsp
14     mov     DWORD PTR [rbp-4], edi
15     mov     eax, DWORD PTR [rbp-4]
16     lea     edx, [rax+3]
17     test    eax, eax
18     cmovs   eax, edx
19     sar     eax, 2
20     pop     rbp
21     ret
22  .LC0:
23     .string "abcdef"
24  main:
25     push    rbp
26     mov     rbp, rsp
27     sub     rsp, 32
28     mov     QWORD PTR [rbp-8], OFFSET FLAT:.LC0
29     mov     DWORD PTR [rbp-12], 11148
30     mov     eax, DWORD PTR [rbp-12]
31     lea     edx, [rax+7]
32     test    eax, eax
33     cmovs   eax, edx
34     sar     eax, 3
35     mov     DWORD PTR [rbp-16], eax
36     mov     eax, DWORD PTR [rbp-12]
37     lea     edx, [rax+3]
38     test    eax, eax
39     cmovs   eax, edx
40     sar     eax, 2
41     mov     DWORD PTR [rbp-20], eax
42     mov     eax, DWORD PTR [rbp-12]
43     mov     edx, eax
44     shr     edx, 31
45     add     eax, edx
46     sar     eax
47     mov     DWORD PTR [rbp-24], eax
48     mov     esi, 4
49     mov     edi, 5
50     call    div(int, int)
51     mov     DWORD PTR [rbp-28], eax
52     mov     edi, 5
53     call    div4(int)
54     mov     DWORD PTR [rbp-32], eax
55     mov     eax, 0
56     leave
57     ret

```

```

1  int div(int x, int y){
2      return x/y;
3  }
4  int div4(int x){
5      return x/4;
6  }
7  int main(){
8      char* c = "abcdef";
9      int m = 11148;
10     int x = m/8;
11     int y = m/4;
12     int z = m/2;
13     int rpt = div(5,4);
14     int rpt2 = div4(5);
15     return 0;
16 }

```

a) En que parte del código ensamblador se define la función div.

```
1  v div(int, int):  
2      push    rbp  
3      mov     rbp, rsp  
4      mov     DWORD PTR [rbp-4], edi  
5      mov     DWORD PTR [rbp-8], esi
```

b) En que parte del código ensamblador se invoca a la función div.

```
48      mov     esi, 4  
49      mov     edi, 5  
50      call    div(int, int)  
51      mov     DWORD PTR [rbp-28], eax
```

c) En que parte del código ensamblador dentro de la función div se procesa la división.

```
6      mov     eax, DWORD PTR [rbp-4]  
7      cdq  
8      idiv    DWORD PTR [rbp-8]
```

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 que se debe dicha diferencia y comente cuales podrian ser las consecuencias. (4 puntos)