Trabajo Práctico Nº 6 A
LOS LENGUAJES Y LA MÁQUINA

Assembler

1) Dado el siguiente código

```
            .org 2048
            .global   main
            .extern   MiFun
cte22       .equ      280000h
cte10       .equ       010h
main:       sethi     cte22, %r12
            add       %r12, cte10, %r12
            add       %r15, %r0, %r16
set:        st        %r0, %r12
            call      MiFun
            orcc      %r2, %r2, %r0
            bneg      f in
            addcc     %r2, -1, %r2
            bne       set
fin:        jmpl      %r16 + 4, %r0
            .org      A00004h
Disp: .dwb  1
```

(a) Indicar la tabla de símbolos incluyendo relocalización y referencias externas,
(b) Señalar cuales son las líneas de este código que no tendrán representación en RAM cuando el programa sea cargado para su ejecución.

2) Idem problema anterior tomando el código que ha propuesto como respuesta a los problemas 22 y 23 de la tira n°5

3) Para las siguientes sentencias en C encontrar las instrucciones en lenguaje simbólico de ARC

for (i = 0: i < N; i++) sentencia

while ((c = getchar()) == ‘ ‘ || c == ‘\n’ || c == ‘\t’)

(a) Escribir su equivalente en lenguaje simbólico de ARC
(b) Obtener el código de máquina y la tabla de símbolos.

4) Dados:

```
.begin
.org 2048
main: or     %r0, %r15, %r16
      ld     %r14, %r1
      call   separa
      st     %r2, %r14
      add    %r14, -4, %r14
      st     %r3, %r14
      jmpl   %r16, 4, %r0
separa: sra  %r1, 16, %r2
      sll    %r1, 16, %r3
      sra    %r3, 16, %r3
      jmpl   %r15, 4, %r0
.end
```

```
.begin
.org 2048
 .macro separa Reg1, Reg2, Reg3
      sra    Reg1, 16, Reg2
      sll    Reg1, 16, Reg3
      sra    Reg3, 16, Reg3
 .endmacro
main: ld     %r14, %r1
      separa %r1, %r2, %r3
      st     %r2, %r14
      add    %r14, -4, %r14
      st     %r3, %r14
      jmpl   %r15, 4, %r0
.end
```

Se pide: (a) determinar el tamaño en bytes del código objeto generado por el ensamblador en cada uno (b) analizar diferencias en su tiempo de ejecución y medirlas a partir de la cantidad de ciclos de reloj insumidos por cada instrucción según el “time model editor” de ARCtools 2.1.2

5) Dos archivos de Assembly ARC fueron ensamblados independientemente y el ensamblador generó en cada caso los siguientes listados:

A

```
HexLoc    DecLoc     MachWord  Label     Instruction
Comment
                                .         equ 1
                                          .org 2048
                                          .global  subrut
00000800  0000002048 86b0c000  subrut:   orncc  %r3, %r0, %r3
00000804  0000002052 8680e001            addcc  %r3, 1, %r3
00000808  0000002056 81c3e004            jmpl   %r15, 4, %r0
```

B	HexLoc DecLoc MachWord Label Instruction Comment . org 2048 .extern subrut 00000800 0000002048 c4002810 main: ld [2064], %r2 00000804 0000002052 c6002814 ld [2068], %r3 00000808 0000002056 40000000 call subrut 0000080c 0000002060 81c3e004 jmpl %r15, 4, %r0 00000810 0000002064 00000069 x: 2 00000814 0000002068 0000005c y: 10

a) Indicar las tablas de símbolos generadas en cada caso
b) Indicar cuáles son los cambios en el código binario que debe realizar un linker (link editor) para en base a ambos generar un único código objeto ejecutable.
c) Indicar cuáles son los cambios en el código binario que debe realizar un linking-loader para la ejecución del código objeto obtenido en el punto (a) si el sistema operativo asigna al programa un segmento de memoria estática que empieza en la dirección 0E28h

Lenguajes de alto nivel

6) Ordenar los siguientes códigos en función de su velocidad de ejecución

A	B	C	D
#define Largo = 25; #define Ancho =10; int main () {int Vol=6*Ancho*Largo;}	#define Cte1 = 25; int main () { int X=6*Cte1*10;}	#define Largo = 25; #define Ancho =10; #define Sup=Ancho*Alto; int main() {int Vol=6*Sup;}	#define Cte1 = 25; #define Cte2 =B001A012h int main () { int X=6*Cte1*Cte2;}

7) Idem ejercicio 6 con:

A	B	C	D

int Largo = 25;
byte Ancho =10;
 	
int main ()
{ int Sup=Ancho*Largo;}
	int Largo = 25;
	int Ancho =10;
	
	int main ()
	{ int Sup=Ancho*Largo;}
		float Largo = 25;
		int Ancho =10;
		
		int main ()
		{int Sup=Ancho*Largo;}
			float Largo = 25;
			int Ancho =10;
			
			int main ()
			{float Sup=Ancho*Largo;}

8) Idem ejercicio 6 con:

A	B	C	D

double Largo = 25;
int Ancho =10;
 	
int main ()
{doublé Sup=Ancho*Largo;}
	float Largo = 25;
	double Ancho =10;
	
	int main ()
	{double Sup=Ancho*Largo;}
		float Largo = 25;
		float Ancho =10;
		
		int main ()
		{double Sup=Ancho*Largo;}
			double Largo = 25;
			double Ancho =10;
			
			int main ()
			{double Sup=Ancho*Largo;}

9) Los siguientes códigos son equivalentes en su función pero difieren en su velocidad de ejecución. Justificar esa diferencia y en base a ello discutir costos y ventajas comparativas de utilizar funciones y procedimientos particularmente en caso de que estén anidados.

A	B

long A=1;
long B=2;
long C;
 	
int main ()
{ A=C*(A+B);}
	long A=1;
	long B=2;
	long C;
	long Suma(long x, long y)
	{ return x+y; }
	int main ()
	{ A=C*Suma(A,B); return 0}