Rubén Dario A O

2) -la simplicidad favorece la regularidad: son instrucciones triadicas para operaciones aritméticas
- pequeño es mas rápido: dependiendo de la implementación es la cantidad numero de registros como son limitados debemos usar la cantidad menor posible
-hacer el caso común primero: podemos determinar mediante la codificación de una instrucción cuando usar constantes y cuando usar constantes y cuando registros. Podemos realizar incrementos
- grandes diseños demandan grandes compromisos: debido a su cantidad de instrucciones que están en tres grandes categorías se nota el tiempo de desarrollo de diseño es alto y requiere gran conocimientos y necesitan pruebas

3) Convertir a instrucciones de bajo nivel.

```
int x=0;                                    l1=x
int y =8;                                   l2=y
int z = 1;                                  l3=z
                                            l4

y=x+3;
z=z+3;
x=(x-z)+(3+y);

add %g0, 0 ,%l1
add %g0, 8 ,%l2
add %g0, 1 ,%l3
add %l1, 3 ,%l2
add %l3, 3 ,%l4
sub %l1, % l4 ,%l5
add %l2 , % 3 , % l6
add %l5 , %l6
```

10 | 10001 | 000000 | 00000 | 1 | 0000000000000
10 | 10010 | 000000 | 00000 | 1 | 0000000001000
10 | 10011 | 000000 | 00000 | 1 | 0000000000001
10 | 10010 | 000000 | 10001 | 1 | 0000000000011
10 | 10100 | 000000 | 10011 | 1 | 0000000000011
10 | 10101 | 000100 | 10001 | 0 | 00000000 | 10100
10 | 10110 | 000000 | 10010 | 1 | 0000000000011
10 | 10001 | 000000 | 10101 | 0 | 0000000010110

4. Usar el ld, y st.
a[4]= a[2]+x;

l1=a l2=x

```
or %g0 , 4 ,%l1
st %o1 , l1 , 28
add %l1 ,%l2, %l1
```

y = y[40]+13;

or %g0 , 13 , %l2
st %g0, 40 , %l1
add %l1, %l2 , %l1

5. Convertir a lenguaje de maquina.
a.

```
int main(){

        int i =3; p=2;
        return i+3;
}
```

l1=i
l2=p

add %g0 , 3 , %l1
add %g0 , 2 , %l2
add %l1 , 3 , %l4
add %g0 , %l4 , %o0

10 | 10001 | 000000 | 000000 | 1 | 0000000000011
10 | 10001 | 000000 | 000000 | 1 | 0000000000010
10 | 10100 | 000000 | 010001 | 1 | 0000000000011
10 | 10000 | 000000 | 000000 | 0 | 00000000 | 10100

b.

```
int main(){
        int p=3; x=1; z=4;
        int w=0;
        w=(p+40)+(x-z);
        return 0;
}
```

l1=p
l2=x
l3=z

l4=w

```
add %g0 , 3 , % l1
add %g0 , 1 , % l2
add %g0 , 4 , %l3
add %g0 , 0 , %l4
add %l1 , 40 , %l5
sub %l2 , %l3 , %l6
add %l5, %l6 , %l4
add %g0 ,%l4, %o0
```

```
10 | 10001 | 000000 | 00000 | 1 | 0000000000011
10 | 10010 | 000000 | 00000 | 1 | 0000000000001
10 | 10011 | 000000 | 00000 | 1 | 0000000000100
10 | 10100 | 000000 | 00000 | 1 | 0000000000000
10 | 10101 | 000000 | 10001 | 1 | 0000000101000
10 | 10110 | 000100 | 10010 | 0 | 00000000 | 10011
10 | 10101 | 000000 | 10100 | 0 | 00000000 | 10110
10 | 10000 | 000000 | 00000 | 0 | 00000000 | 10100
```

6. Inicializar las siguientes variables negativas usando OR.

n=-12,
a=-11,
b=-14.

```
Or %g0, -12, l1
or %g0, -11,%l2
or %g0, -14,%l3
```