

Taller #001 - Febrero de 2016

1. Escriba los 4 principios de diseño de hardware aprendidos en clase.

- 1. La simplicidad favorece la regularidad
- 2. Entre más pequeño, más rápido
- 3. Hacer el caso común más rápido
- 4. Buenos diseños demandan buenos compromisos

2. Convertir a instrucciones de bajo nivel.

```
int x = 0;
int y = 8;
int z = 1;
y=x+3;
z=z+3;
x=(x-z)+(3+y);
# Inicializo las variables
add %g0, 0, %l0 ; # x = 0
add %g0, 8, %l1 ; # y = 8
add %g0, 1, %l2 ; # z = 1
```

opcode	op	rd	op3	rs1	i	simm13	hex
add	10	10000	000000	00000	1	0000000000000	A0002000
add	10	10001	000000	00000	1	0000000001000	A2002008
add	10	10010	000000	00000	1	0000000000001	A4002001

```
# Procedo con las operaciones
add %l0, 3, %l1 ; # y = x + 3;
add %l2, 3, %l2; # z = z + 3;
sub %l0, %l2, %l0 ; # x = x - z;
add %l1, 3, %l3 ; # tmp = 3 + y;
add %l0, %l3, %l0 ; # x = x + tmp;
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2		hex
add	10	10001	000000	10000	1	0000000000011		A2042003
add	10	10010	000000	00010	1	0000000000011		A400A003
sub	10	10000	000100	10000	0	00000000	10010	A0240012
add	10	10011	000000	10001	1	0000000000011		A6046003
add	10	10000	000000	10000	0	00000000	10011	A0040013

```
3. Usar el ld, y st.
a[4]= a[2]+x;y = y[40]+13;
# %l0 contiene a[]
# %l1 contiene y[]
add %g0, 0, %l3 ; # x = 0
ld [%l0 + (2*4)], %l 4; # cargo el contenido de a[2] en %l4
add %l3, %l4, %l4; # %l4 = a[2] + x
st %l4, [%l0 + (4*4)] ; # guardo el contenido de %l4 en a[4]
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
add	10	10011	000000	00000	1	00000000000000	A6002000
ld	11	10100	000000	10000	1	0000000001000	E8042008
add	10	10100	000000	10011	0	0000000010100	A804C014
st	11	10100	000100	10000	1	0000000010000	E8242010

```
ld [%l1 + (40*4)], %l 5; # cargo el contenido de y[40] en %l5
add %l5, l3, %l1 ; # y = %l5 + l3
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
ld	11	10101	000000	10001	1	0000010100000	EA0460A0
add	10	10001	000000	10101	1	0000000001101	A205600D

4. Convertir a lenguaje de mÃquina.

```
a.
int main(){
  int i =3; p=2;
  return i+3;
}
add %g0, 3, %g 1; # i = 3
add %g0, 2, %g 2; # p = 2
add %g1, 3, %o 0; # return i + 3
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
add	10	00001	000000	00000	1	00000000000011	82002003
add	10	00010	000000	00000	1	00000000000010	84002002
add	10	01000	000000	00001	1	00000000000011	90006003

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

```
return 0;
}
```

Inicializo las variables

```
add %g0, 3, %g 1; # p = 3
add %g0, 1, %g2 ; # x = 1
add %g0, 4, %g 3; # z = 4
add %g0, 0, %g 4; # w = 0
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
add	10	00001	000000	00000	1	00000000000011	82002003
add	10	00010	000000	00000	1	00000000000001	84002001
add	10	00011	000000	00000	1	00000000000100	86002004
add	10	00100	000000	00000	1	00000000000000	88002000

Procedo con las operaciones

```
add %g1, 40, %l0; # %l0 = p + 40
sub %g2, %g3, %l1; # %l1 = x - z
add %l0, %l1, %g4; # w = %l0 + %l1
add %g0, 0, %o0 ; # return 0
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
add	10	10000	000000	00001	1	0000000101000	A0006028
sub	10	10001	000100	00010	1	0000000000011	A220A003
add	10	00100	000000	10000	1	00000000010001	88042011
add	10	01000	000000	00000	1	00000000000000	90002000



5. Inicializar las siguientes variables negativas usando OR.

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

```
or %g0, -12, %l0 ; # n = -12
or %g0, -11, %l1 ; # a = -11
or %g0, -14, %l2 ; # b = -14
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2	hex
or	10	10000	000010	00000	1	1111111110100	A0103FF4
or	10	10001	000010	00000	1	1111111110101	A2103FF5
or	10	10010	000010	00000	1	1111111110010	A4103FF2