Taller #001 - Febrero de 2016 - v2

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

int y = 8;

4. Buenos diseños demandan buenos compromisos

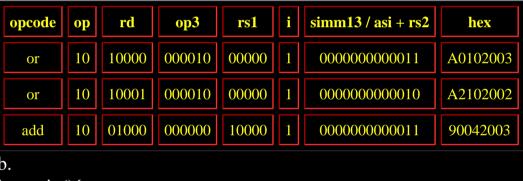
```
2. Convertir a instrucciones de bajo nivel. int x=0;
```

```
int z = 1;
# Inicializo las variables
or \% g0, 0, \%10; # x = 0
or \%g0, 8, \%11; \#y = 8r \%g0, 1, \%12; \#z = 1
y=x+3;
z=z+3;
x=(x-z)+(3+y);
add %10, 3, %11; \# y = x + 3
add %12, 3, %12; \# z = z + 3
sub %10, %12, %13; \# %13 = x - z (tmp)
add %11, 3, %10; \# x = 3 + y
add %10, %13, %10; \# x = (x - z) + (3 + y)
3. Usar el ld, y st.
a[4] = a[2] + x;
y = y[40]+13;
# Presuponemos a[] = \%10, y x = \%11
ld [\%10 + (2 * 4)], \%12; #tmp1 = a[2]
```

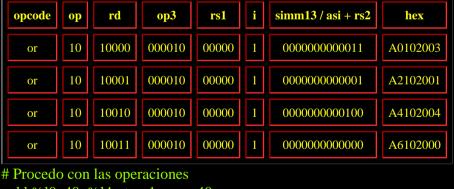
4. Convertir a lenguaje de máquina.

add %12, %11, %12; # tmp1 = a[2] + x st %12, [%10 + (4 * 4)]; # a[4] = a[2] + x

```
int main(){
  int i =3; p=2;
  return i+3;
}
  or %g0, 3, %10; # i = 3
  or %g0, 2, %11; # p = 2
  add %10, 3, %00; # return i + 3
```



```
b.
int main(){
  int p=3; x=1; z=4;
  int w=0;
  w=(p+40)+(x-z);
  return 0;
}
# Inicializo las variables
  or %g0, 3, %l0; # p = 3
  or %g0, 1, %l1; # x = 1
  or %g0, 4, %l2; # z = 4
  or %g0, 0, %l3; # w = 0
```



```
# Procedo con las operaciones
add %10, 40, %14; tmp1 = p + 40
sub %11, %12, %15; tmp2 = x + z
add %14, %15, %13; w = (p+40) + (x-z)
```

opcode	op	rd	op3	rs1	i	simm13 / asi + rs2		hex
add	10	10100	000000	10000	1	0000000101000		A8042028
sub	10	10101	000100	10001	0	00000000	10010	AA244012
add	10	10011	000000	10100	0	00000000	10101	A6050015

5. Inicializar las siguientes variables negativas usando OR. n=-12,

```
a=-11,
b=-14.
or %g0, -12, %10; # n = -12
or %g0, -11, %11; # a = -11
or %g0, -14, %12; # b = -14
```