## Load y Store

```
ld %r1, %r2, %r3 !cargo %r1+%r2 en %r3
st %r1, %r2, %r3 !guardo %r1 en %r2+%r3
```

## Macros de push y pop de stack

```
.macro push arg
add %r14, -4, %r14
st arg, %r14
.end macro

.macro pop arg
ld %r14, arg
add %r14, 4, %r14
.end macro
```

## Arreglos

```
.dwb cant_elementos !declaración
largo: cant_elementos * 4
```

## Lectura de los elementos

```
bucle:      andcc %r2, %r2, %r0
            be fin_array
            add %r2, -4, %r2 !recordar que los arreglos los recorro
de atrás para adelante
            ld %r1, %r2, %r3
            hago algo
            ba bucle
!%r2: largo del array
!%r1: inicio del array, si el array está declarado se reemplaza por
[array]
!%r3: donde guardo el elemento leído
```

## Escritura de elementos en un array

```
aniadir_elem:    add %r4, -4, %r4
                 add %r4, [array], %r3
                 st %r2, %r3
```

u otra versión:

```
aniadir_elem:    add %r4, -4, %r4
                 st %r2, %r4, [array]
!%r4:largo
![array]: inicio del array
!%r2: elemento a guardar
```

## Dispositivos/Periféricos

### Lectura

```
ld[periferico], %r1 !guardo dirección del dispositivo
ld %r1, %r2 !cargo en %r2 el contenido de la dirección %r1
periferico: 0xA3250000
```

### Escritura

```
ld[periferico], %r1 !guardo dirección del dispositivo
st %r2, %r1 !guardo %r2 en la dirección %r1
periferico: 0xA3250000
```

### Cómo hacer multiplicaciones y divisiones (si me piden promedio por ej.)

Si por lo que tengo que multiplicar o dividir es potencia de 2, (1, 2, 4, 8, 16, 32, 64…), utilizo desplazamientos.

Voy a desplazar el exponente de la potencia de 2, si es 16, desplazo 4 (2 a la 4 = 16)

Si quiero dividirlo desplazo hacia la DERECHA, puedo usar SRL.
Si quiero multiplicarlo desplazo hacia la IZQUIERDA, puedo usar SLL.