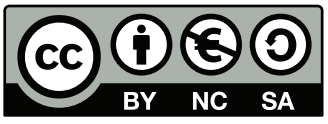
Estructura de Computadores

LibreIM

Doble Grado de Informática y Matemáticas
Universidad de Granada

libreim.github.io/apuntesDGIIM



Estructura de Computadores

LibreIM

Doble Grado de Informática y Matemáticas
Universidad de Granada

libreim.github.io/apuntesDGIIM

Índice

1 Estructura de computadores. Tema Introducción.

1.1 Estructuras de bus

Existen:

- Bus único: Barato, sencillo pero muy ineficiente
- Bus múltiple: es un bus de sistema que comunica CPU-Memoria

2 Tema 2

2.1 Estado visible al programador:

- Registros
- Contador del programa:PC.
- Registro de condición: cada bit tiene un significado de estad.

También un programador necesita ver: * Si la memoria es direccionable por bytes * Dónde está el inicio de la pila

2.1.1 Ejemplo de trabajo

Código en c:

```
1 int sum(int x,int y)
2 {
3   int t = t+y
4   return t;
5 }
```

Código en ensamblador IA32:

```
1 push1 %ebp
2 mov1 %esp,%ebp
3 mov1 12(%ebp)
4 addl 8(%ebp),%eax
5 pop1 %ebp
```

- %eax es el el regístro que conteiene la última operación hecha. Almacena primero registros en la pila con el push, luego mueve los registros y los almacenos en otros registros y los manipula haciendo la operación y hago devuelvo los registros originales mediante el pop.

2.2 Características del ensamblador:

2.2.1 Datos

- Los datos **enteros** son de 1, 2 ó 4 bytes. Son los que utilizaremos
- Existen los datos en punto flotante pero no los daremos
- No hay arrays ni Estructuras

2.2.2 Operaciones

- De función aritmética
- De transferencia de datos entre memoria
- De control

2.2.3 Ejemplo

3 Instrucciones en Ensamblador

push: Decrementa el puntero de pila (ESP) el número de posiciones de memoria que ocupe el dato a insertar, posteriormente procede a escribir el dato en esas posiciones reservadas, a partir de donde apunta ESP ahora

```
1 push %edx
```

pop: Lee el tope de la pila, guardando el valor de esa dirección donde indique el argumento, posteriormente incrementa el puntero de pila (ESP)

```
1 pop %edx
```

call: Guarda la dirección de retorno en la pila antes de saltar a la subrutina indicada como argumento

```
1 call suma     (llama a la subrutina etiquetada como "suma")
```

ret: Recupera de la pila la dirección de retorno

```
1 ret
```

mov: Mueve el src al dest

```
1 mov $0,%eax
```

add: Suma al registro de destino el src

```
1 add (%ebx,%ebx,4),%eax
2
3 (Con el formato (%ebx,%edx,4) %ebx es el registro base, al
    cual se le suma el contenido de %edx, multiplicado por 4 y
    se añade en %eax)
```

inc: Incrementa en una unidad el registro indicando

```
1 add %edx
```

cmp: Compara el contenido de dos registros

jne: Salta a la subrutina indicada por la etiquetada si