Laboratorio ARM

Organización del Computador

Parte 2

Temas

Instrucción Compare (cmp)

Instrucciones ejecutadas condicionalmente

Sentencias condicionales simples y complejas

Loops

Recursividad

Instrucción Compare (cmp)

Compare (cmp)

Resta dos operandos y descarta el resultado.

Sin embargo, se setean los bits de estado (por ej.: acarreo, cero, etc.).

Compare (cmp)

Resta dos operandos y descarta el resultado.

Sin embargo, se setean los bits de estado (por ej.: acarreo, cero, etc.).

```
mov     r0,     #5
cmp     r0,     #5
```

Compare (cmp)

Resta dos operandos y descarta el resultado.

Sin embargo, se setean los bits de estado (por ej.: acarreo, cero, etc.).

```
mov     r0,     #5
cmp     r0,     #5
```

Setea bit/flag cero (resultado es cero)

Compare (cmp)

Resta dos operandos y descarta el resultado.

Sin embargo, se setean los bits de estado (por ej.: acarreo, cero, etc.).

```
mov    r0,    #5
cmp    r0,    #5
```

Setea bit/flag cero (resultado es cero)

```
mov    r0,    #5
cmp    r0,    #20
```

Compare (cmp)

Resta dos operandos y descarta el resultado.

Sin embargo, se setean los bits de estado (por ej.: acarreo, cero, etc.).

```
mov    r0,    #5
cmp    r0,    #5
```

Setea bit/flag cero (resultado es cero)

```
mov    r0,    #5
cmp    r0,    #20
```

Setea bit/flag negativo (resultado es negativo)

Significado

Los bits de estado dicen algo sobre el resultado de las comparaciones aritméticas

Significado

Los bits de estado dicen algo sobre el resultado de las comparaciones aritméticas

---

```
mov     r0,     #5
cmp     r0,     #5
```

Setea bit/flag cero (resultado es cero)

Significado

Los bits de estado dicen algo sobre el resultado de las comparaciones aritméticas

---

Los operandos son iguales

```
mov     r0,     #5
cmp     r0,     #5
```

Setea bit/flag cero (resultado es cero)

Significado

Los bits de estado dicen algo sobre el resultado de las comparaciones aritméticas

Los operandos son iguales	`mov r0, #5` `cmp r0, #5`	Setea bit/flag cero (resultado es cero)
	`mov r0, #5` `cmp r0, #20`	Setea bit/flag negativo (resultado es negativo)

Significado

Los bits de estado dicen algo sobre el resultado de las comparaciones aritméticas

Los operandos son iguales	`mov r0, #5` `cmp r0, #5`	Setea bit/flag cero (resultado es cero)
1° operando < 2° operando	`mov r0, #5` `cmp r0, #20`	Setea bit/flag negativo (resultado es negativo)

Ejecución condicional

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

```
movmi  r0, #42
```

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

```
movmi   r0, #42
```

mover si el bit negativo está encendido

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

```
movmi   r0, #42
movpl   r0, #42
```

mover si el bit negativo está encendido

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

```
movmi   r0, #42
```
mover si el bit negativo está encendido

```
movpl   r0, #42
```
mover si el bit negativo **no** está encendido

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

```
movmi   r0, #42
movpl   r0, #42
moveq   r0, #42
```

mover si el bit negativo está encendido

mover si el bit negativo **no** está encendido

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

`mov`**`mi`** `r0, #42`	mover si el bit negativo está encendido
`mov`**`pl`** `r0, #42`	mover si el bit negativo **no** está encendido
`mov`**`eq`** `r0, #42`	mover si el bit cero está encendido

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

`mov`**`mi`** `r0, #42`	mover si el bit negativo está encendido
`mov`**`pl`** `r0, #42`	mover si el bit negativo **no** está encendido
`mov`**`eq`** `r0, #42`	mover si el bit cero está encendido
`mov`**`ne`** `r0, #42`	

Ejecución condicional

ARM permite que las instrucciones se ejecuten *condicionalmente* a los valores de los bits de estado

`mov`**`mi`** `r0, #42`	mover si el bit negativo está encendido
`mov`**`pl`** `r0, #42`	mover si el bit negativo **no** está encendido
`mov`**`eq`** `r0, #42`	mover si el bit cero está encendido
`mov`**`ne`** `r0, #42`	mover si el bit cero **no** está encendido

Demo

Ejecución condicional

Práctica 7

Cálculo de valor absoluto con instrucciones condicionales

Instrucción Branch (b)

Instrucción Branch (b)

Branch (b) se usa para saltar a código marcado con una etiqueta.
El código puede ser etiquetado, al igual que los datos.

Instrucción Branch (b)

Branch (b) se usa para saltar a código marcado con una etiqueta.
El código puede ser etiquetado, al igual que los datos.

```
    mov r0, #1
    b otra
    mov r0, #5
otra:
    mov r1, r0
```

Instrucción Branch (b)

Branch (b) se usa para saltar a código marcado con una etiqueta.
El código puede ser etiquetado, al igual que los datos.

```
    mov r0, #1
    b otra
    mov r0, #5
otra:
    mov r1, r0
```

-La ejecución de branch hace que la ejecución salte a *otra*.

-La instrucción *mov r0, #5* nunca se ejecuta.

Branch condicional

Punto clave: la instrucción b puede ser ejecutada condicionalmente

Branch condicional

Punto clave: la instrucción b puede ser ejecutada condicionalmente

```
    mov r0, #0
    mov r1, #5
    cmp r1, #5
    beq otrolado
    mov r0, #25
otrolado:
    mov r2, r0
```

Branch condicional

Punto clave: la instrucción b puede ser ejecutada condicionalmente

```
        mov r0, #0
        mov r1, #5
        cmp r1, #5
        beq otrolado
        mov r0, #25
otrolado:
        mov r2, r0
```

-Hace 5 – 5 y enciende el bit cero
-Porque el bit cero está encendido: se produce el salto

Ejecución condicional & Branch

- Para traducir sentencias condicionales complejas es más conveniente usar branches etiquetados
- Requerido para condiciones anidadas
- Las instrucciones ejecutadas condicionalmente son más útiles para condiciones simples
    - En general: hasta tres instrucciones

Práctica 8

Cálculo de valor absoluto con bifurcación

Práctica 9

Cálculo de mínimo y máximo

Loops

Loops

Se utiliza la instrucción Branch con saltos al inicio o al final de un loop

Práctica 11

Imprimir los números del 0 al 9

Práctica 12

Cálculo de factorial

Práctica 13

Cálculo recursivo de factorial

Revisión de temas

Instrucción Compare (cmp)

Instrucciones ejecutadas condicionalmente

Sentencias condicionales simples y complejas

Loops

Recursividad

Laboratorio ARM

Organización del Computador