Tema 2. Funcionamiento del computador
3. Estudio del Computador Básico

http://lorca.act.uji.es/ig09/

Sergio Barrachina **Germán Fabregat** **Rafael Mayo**

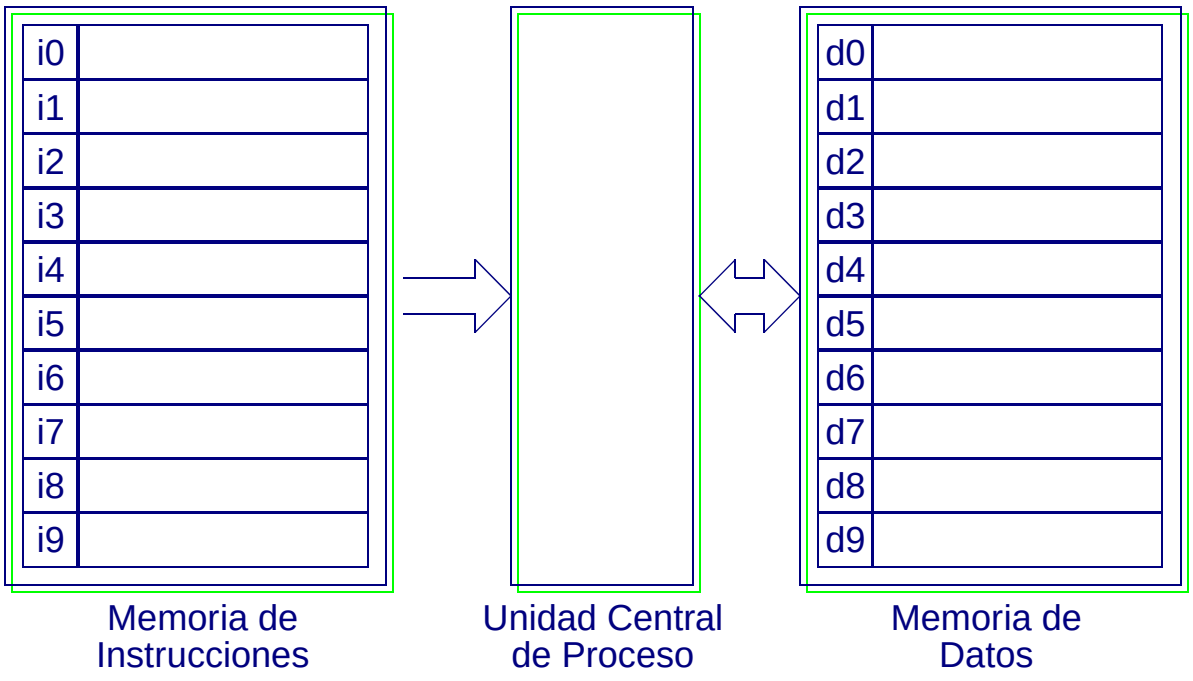
Índice General

1 Arquitectura y organización

Contenido

- Bloques funcionales del Computador Básico
- Conjunto de instrucciones

1.1 Bloques funcionales del Computador Básico



1.2 Conjunto de instrucciones

- Instrucciones de transformación

Instrucción	Acción
`suma d0,d1,d2`	$d0 = d1 + d2$
`resta d0,d1,d2`	$d0 = d1 - d2$
`sumac d0,d1,num`	$d0 = d1 + num$
`restac d0,d1,num`	$d0 = d1 - num$

1.2 Conjunto de instrucciones (II)

- Instrucciones de transferencia

Instrucción	Acción
`movec d0,num`	$d0 = num$

1.2 Conjunto de instrucciones (III)

- Instrucciones de control de flujo

Instrucción	Acción
`salta i5`	$sigInstr = i5$
`salta= d0,d1,i5`	si $d0 == d1 \Rightarrow sigInstr = i5$
`salta> d0,d1,i5`	si $d0 > d1 \Rightarrow sigInstr = i5$
`salta< d0,d1,i5`	si $d0 < d1 \Rightarrow sigInstr = i5$

- Instrucciones especiales

Instrucción	Acción
`fin`	Detiene la ejecución

2 Programación del computador

Contenido

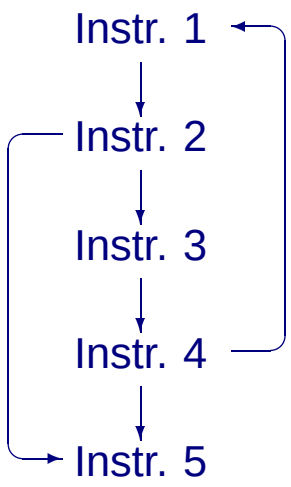
- Modelo de ejecución
- Ejemplos
- Campos de una instrucción
- Modos de direccionamiento

2.1 Modelo de ejecución (y de programación)

Secuencia:

- Se lee una instrucción (de la memoria)
- Se ejecuta
- Se continua con la siguiente instrucción.

Flujo de programa → siguiente instrucción



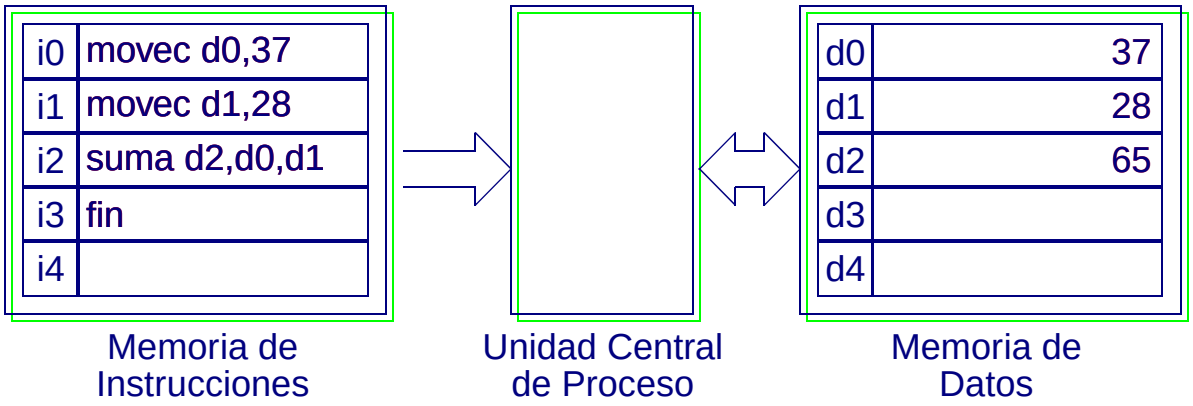
2.2 Ejemplos

Problema: Hallar la suma de los números 37 y 28.
Solución:

- Algoritmo (datos: 37, 28 y $suma$):
  1. $sumando1 = 37$
  2. $sumando2 = 28$
  3. $suma = sumando1 + sumando2$
  4. fin
- Programa (en ensamblador del *Computador básico*):

i0	`movec d0,37`
i1	`movec d1,28`
i2	`suma d2,d0,d1`
i3	`fin`

2.2 Ejemplos (II)



Paso 1: Cargar el programa en la memoria de instrucciones.

Paso 2: Ejecutar la primera instrucción: instrucción i0.

Paso 3: Ejecutar la siguiente instrucción: instrucción i1.

Paso 4: Ejecutar la siguiente instrucción: instrucción i2.

Paso 5: Ejecutar la siguiente instrucción: instrucción i3.

—Finalización de la ejecución —

2.2 Ejemplos (III)

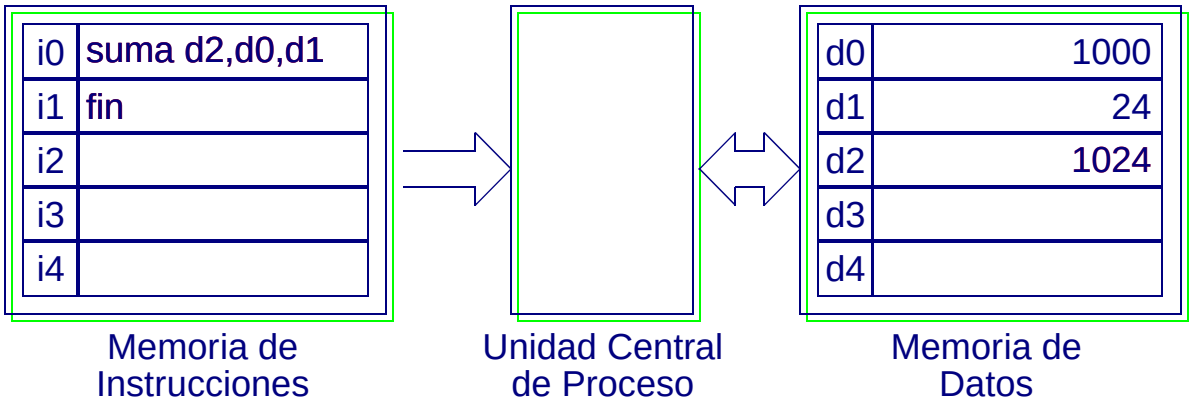
Problema: Hallar la suma de dos números cualesquiera.

Solución:

- Algoritmo (datos: $sumando1$, $sumando2$ y $suma$):
  1. $suma = sumando1 + sumando2$
  2. fin
- Programa (en ensamblador del *Computador básico*):
  - $sumando1$ está en la posición $d0$,
  - $sumando2$ en la posición $d1$ y
  - $suma$ se almacena en la posición de memoria $d2$

i0	`suma d2,d0,d1`
i1	`fin`

2.2 Ejemplos (IV)



Paso 1: Cargar el programa en la memoria de instrucciones.

Paso 2: Cargar los datos en la memoria de datos.

Paso 3: Ejecutar la primera instrucción: instrucción i0.

Paso 4: Ejecutar la siguiente instrucción: instrucción i1.

—Finalización de la ejecución —

2.2 Ejemplos (V)

Problema: Multiplicación de dos números (sin operación de multiplicación).

Solución:

- Algoritmo (datos: $factor1$, $factor2$ y $producto$):
  1. $producto = 0$
  2. $producto = producto + factor2$
  3. $factor1 = factor1 - 1$
  4. si $factor1 > 0$ ir a 2
  5. fin

¿Es correcto?

¿Siempre?

2.2 Ejemplos (VI)

Problema: Multiplicación de dos números (sin operación de multiplicación).

Solución:

- Algoritmo II (datos: $factor1$, $factor2$ y $producto$):
  1. $producto = 0$
  2. si $factor1 == 0$ ir a 6
  3. $producto = producto + factor2$
  4. $factor1 = factor1 - 1$
  5. si $factor1 > 0$ ir a 3
  6. fin

¿Es eficiente?

2.2 Ejemplos (VII)

- Algoritmo III (datos: $factor1$, $factor2$, $producto$, $contador$, $sumador$):
  1. $contador = factor1$
  2. $sumador = factor2$
  3. si $factor1 < factor2$ ir a 6
  4. $contador = factor2$
  5. $sumador = factor1$
  6. $producto = 0$
  7. si $contador == 0$ ir a 11
  8. $producto = producto + sumador$
  9. $contador = contador - 1$
  10. si $contador > 0$ ir a 8
  11. fin

- Programa (en ensamblador del Computador Básico)

i0	`sumac d2,d0,0`	// Mueve d0 (factor1) a d2 (contador)
i1	`sumac d3,d1,0`	// Mueve d1 (factor2) a d3 (sumador)
i2	`salta< d0,d1,i5`	// Si $factor1 < factor2$ ir a i5
i3	`sumac d2,d1,0`	
i4	`sumac d3,d0,0`	
i5	`movec d4,0`	// $d4(producto) = 0$
i6	`movec d5,0`	// d5 se utiliza para comparar con cero
i7	`salta= d2,d5,i11`	// si $contador == 0$ ir a i11
i8	`suma d4,d4,d3`	
i9	`restac d2,d2,1`	
i10	`salta> d2,d5,i8`	
i11	`fin`	

2.2 Ejemplos (VIII)

Problema: Hallar la división entera de dos números enteros positivos cualesquiera. El divisor no puede ser cero.

Solución:

- Algoritmo (datos: $dividendo$, $divisor$ y $cociente$ y $resto$):
  1. $cociente = 0$
  2. $resto = dividendo$
  3. si $resto < divisor$ ir a 7
  4. $resto = resto - divisor$
  5. $cociente = cociente + 1$
  6. ir a 3
  7. fin

2.2 Ejemplos (IX)

- Programa (en ensamblador del Computador Básico)
  - $d1$ ($dividendo$), $d2$ ($divisor$)
  - $d3$ ($cociente$), $d4$ ($resto$)

i0	`movec d3,0`	
i1	`sumac d4,d1,0`	// Mueve d1 (dividendo) a d4 (resto)
i2	`salta< d4,d2,i6`	
i3	`resta d4,d4,d2`	// resto=resto-divisor
i4	`sumac d3,d3,1`	
i5	`salta< d4,d2,i3`	
i6	`fin`	

2.2 Ejemplos (X)

Problema:

Desarrollar la parte de un videojuego encargada de actualizar la posición de la nave espacial del jugador en función del movimiento que el usuario indique mediante una palanca de mandos.

La posición de la nave se indica mediante el par de coordenadas x e y que tienen su origen en la parte inferior izquierda de la pantalla.

La palanca de mandos utiliza cuatro variables para determinar la posición de la misma: $izquierda$, $derecha$, $arriba$ y $abajo$. Estas variables valen 1 cuando el usuario mueve la palanca en dicha dirección y 0 en caso contrario. Así, si el jugador mueve la palanca arriba y hacia la derecha, las variables valdrán $izquierda = 0$, $derecha = 1$, $arriba = 1$ y $abajo = 0$.

Los incrementos de las coordenadas tienen que ser de 1 en 1.

2.2 Ejemplos (XI)

Solución:

- Algoritmo:
  1. si $arriba == 0$ ir a 3
  2. $y = y + 1$
  3. si $abajo == 0$ ir a 5
  4. $y = y - 1$
  5. si $izquierda == 0$ ir a 7
  6. $x = x - 1$
  7. si $derecha == 0$ ir a 9
  8. $x = x + 1$
  9. fin

2.2 Ejemplos (XII)

Solución:

- Algoritmo (II):
  1. $x = x - izquierda$
  2. $x = x + derecha$
  3. $y = y + arriba$
  4. $y = y - abajo$
  5. fin

2.2 Ejemplos (XIII)

Solución:

- Programa (en ensamblador del Computador Básico)
  - $d1$ (x), $d2$ (y)
  - $d3$ ($izquierda$), $d4$ ($derecha$)
  - $d5$ ($arriba$), $d6$ ($abajo$)

i0	`resta d1,d1,d3`
i1	`suma d1,d1,d4`
i2	`suma d2,d2,d5`
i3	`resta d2,d2,d6`
i4	`fin`

2.2 Ejemplos (XIV)

Problema propuesto: Modificar el anterior programa para evitar que las coordenadas puedan salirse de la pantalla (cuya resolución es de 800×600). Es decir, se tiene que garantizar que $x \in [0 \ldots 800[$ y que $y \in [0 \ldots 600[$.

2.2 Ejemplos (XV)

Problema propuesto: Diseñar un algoritmo que calcule el factorial un número. Una vez realizado el algoritmo, realizar el correspondiente programa en ensamblador del Computador Básico.

2.3 Campos de una instrucción

Cada instrucción está formada por algunos de los siguientes campos:

- Operación: la operación que se quiere realizar.
- Operando(s) fuente(s): dato(s) sobre el(los) que se realiza la operación.
- Operando destino: lugar donde se almacena el resultado.

- Ejemplo: `suma d4,d2,d7`
  - Operación: `suma`
  - Operandos fuente: `d2` y `d7`
  - Operandos destino: `d4`

2.4 Modos de direccionamiento

- La dirección en la que se encuentra los operandos de una instrucción puede indicarse utilizando distintos *modos de direccionamiento*.
- El Computador Básico dispone de los siguientes *modos de direccionamiento*:
  - **Directo a memoria**: se indica la posición de memoria en la que se encuentra el operando.
    Ej. `sumac` **`d1,d4,`**`3045`
  - **Dato inmediato**: el operando se encuentra en la misma instrucción.
    Ej. `sumac d1,d4,`**`3045`**

3 Fases de ejecución

La *UCP* ejecuta cada instrucción completando por orden las siguientes fases:

1. Adquisición de la instrucción e incremento del PC.
2. Lectura de operandos fuente.
3. Ejecución de la operación.
4. Escritura de resultados.

> El PC (o contador de programa) es el lugar donde la CPU guarda la dirección de la siguiente instrucción a ejecutar.