Teoría

¿Qué vimos la clase pasada?

¿Qué vimos la clase pasada?

- Compuertas lógicas:

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  – Compuertas: OR, AND, NOT

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos
  - Formulas y tablas de verdad

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos
  - Formulas y tablas de verdad
  - Producto de sumas y suma de productos

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos
  - Formulas y tablas de verdad
  - Producto de sumas y suma de productos
  - Circuitos comunes

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos
  - Formulas y tablas de verdad
  - Producto de sumas y suma de productos
  - Circuitos comunes
  - Circuitos aritméticos

¿Qué vimos la clase pasada?

- Compuertas lógicas:
  - Compuertas: OR, AND, NOT
  - Otras compuertas: XOR, NAND, NOR
- Circuitos
  - Formulas y tablas de verdad
  - Producto de sumas y suma de productos
  - Circuitos comunes
  - Circuitos aritméticos
  - Circuitos especiales

Modelo de Von Neumann

Modelo de Von Neumann

¿Dónde se guardan las instrucciones?

Modelo de Von Neumann

¿Dónde se guardan las instrucciones?

¿Cómo se guardan las instrucciones?

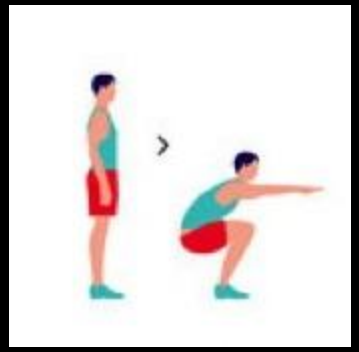
Modelo de Von Neumann

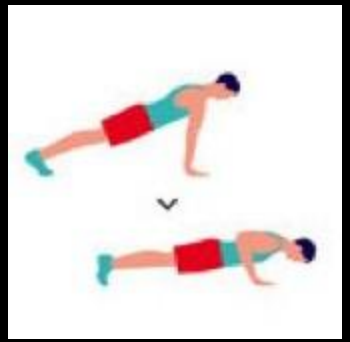
¿Dónde se guardan las instrucciones?

¿Cómo se guardan las instrucciones?

¿Cómo distinguimos cada instrucción al estar en código máquina (binario)?

Rutina de ejercicios





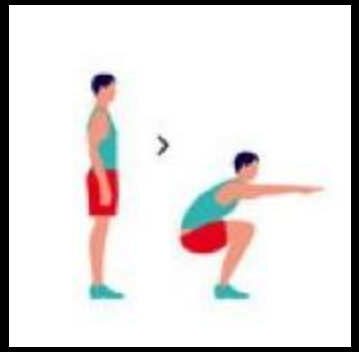
Rutina de ejercicios



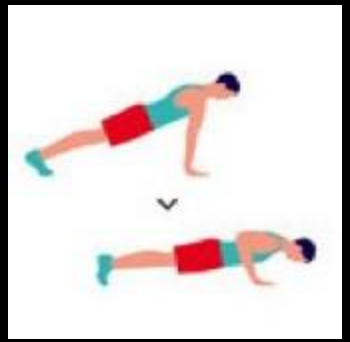


Formato de instrucción

00



01



10

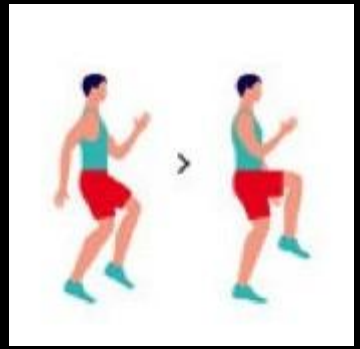
11

Rutina de ejercicios

01

01

00



10

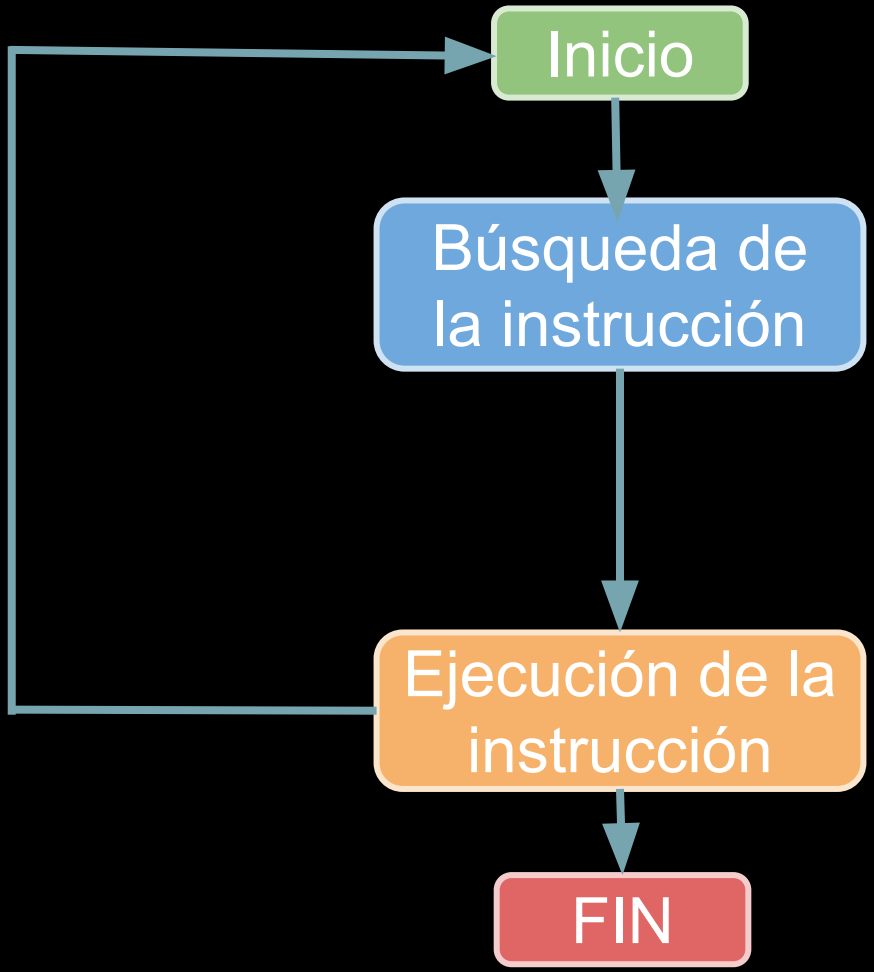
11

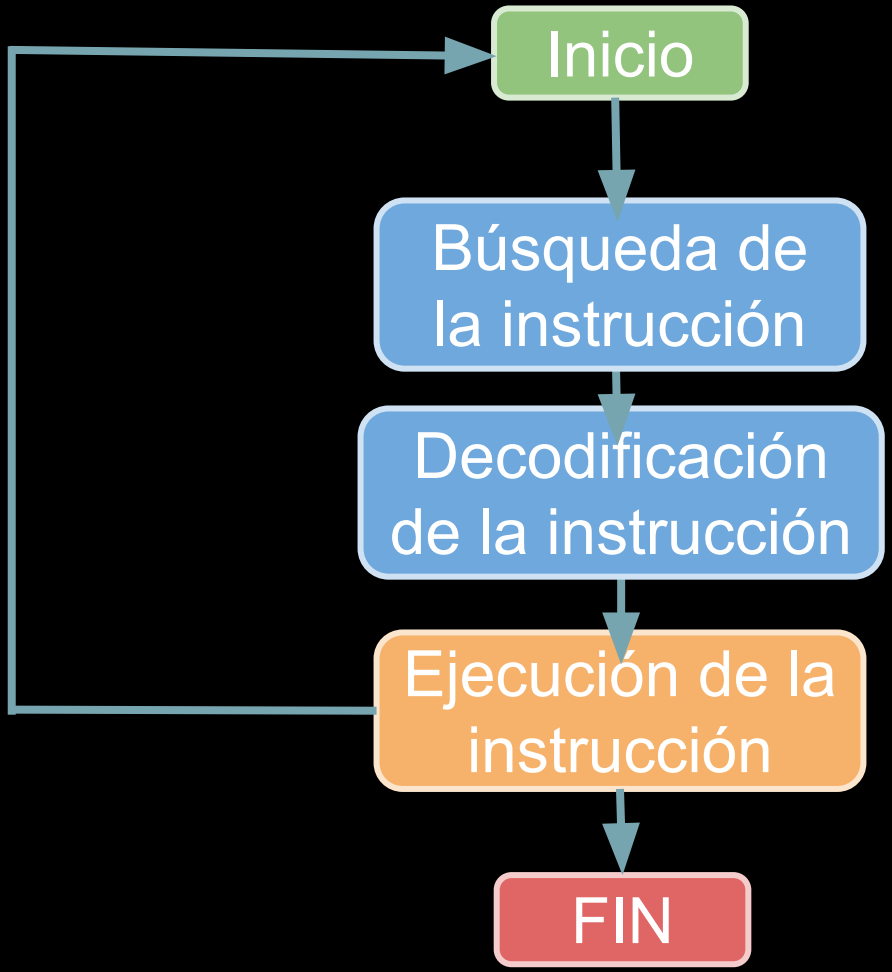
Rutina de ejercicios





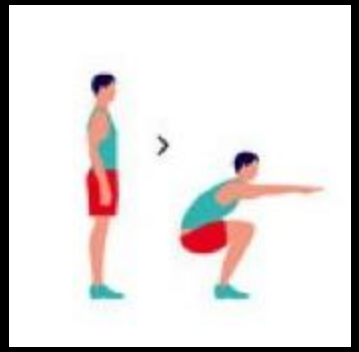
0101001011


Inicio
Búsqueda de la instrucción
Ejecución de la instrucción
FIN

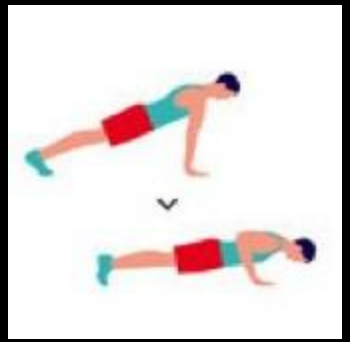

Inicio
Búsqueda de la instrucción
Decodificación de la instrucción
Ejecución de la instrucción
FIN

Rutina de ejercicios

00



01



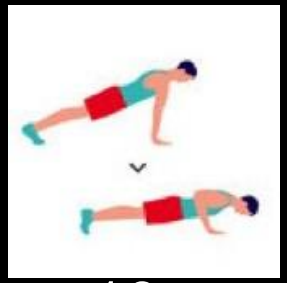
10

1

Rutina de ejercicios

¿1001?

Formato de instrucción



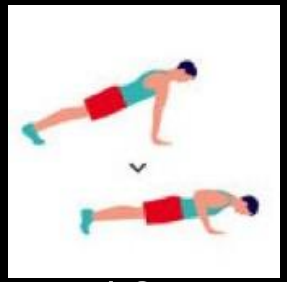
10

01

¿1001?

Formato de instrucción

¿1001?



10

01

1

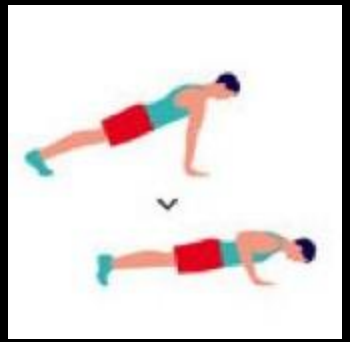
00

1

Formato de instrucción

00

01

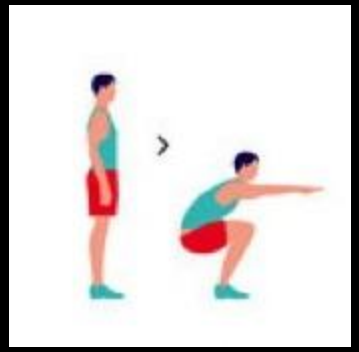


10

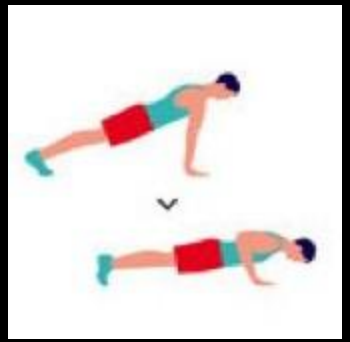
1

Formato de instrucción

00



01



10

11

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

Código fuente

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

Código fuente

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

Código fuente

Código máquina

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

Código fuente

Código máquina

```
1  module(mutex).
2  -export([create/0, lock/1, unlock/1]).
3  create() ->
4      spawn(fun() -> unlocked() end).
5
6  lock(Mutex) ->
7      Mutex ! {lock, self()},
8      receive
9          ok -> ok
10     end.
11
12 unlock(Lock) ->
13     Lock ! {unlock, self()}.
14
15 unlocked() ->
16     receive {lock, LockedBy} ->
17         LockedBy ! ok,
18         locked(LockedBy, 1)
19     end.
```

Código fuente

Ensamblar

Código máquina

```
1   module(mutex).
2   -export([create/0, lock/1, unlock/1]).
3   create() ->
4       spawn(fun() -> unlocked() end).
5
6   lock(Mutex) ->
7       Mutex ! {lock, self()},
8       receive
9           ok -> ok
10      end.
11
12  unlock(Lock) ->
13      Lock ! {unlock, self()}.
14
15  unlocked() ->
16      receive {lock, LockedBy} ->
17          LockedBy ! ok,
18          locked(LockedBy, 1)
19      end.
```

Código fuente

Ensamblar

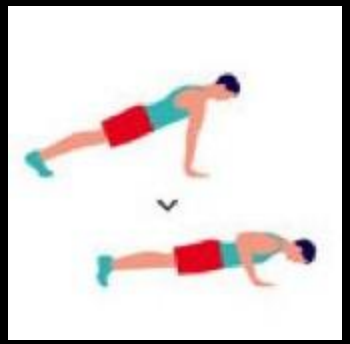
Desensamblar

Código máquina

Desensamblar

00

01



10

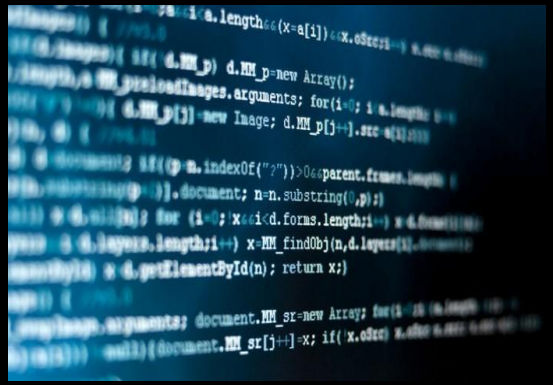
11

- 0000111001
- 1100110110

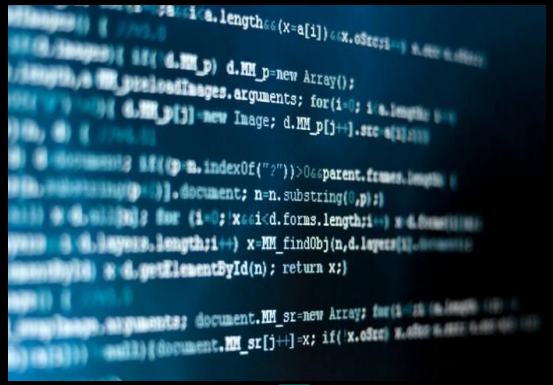
Ciclo de vida de un programa

Ciclo de vida de un programa

Ciclo de vida de un programa

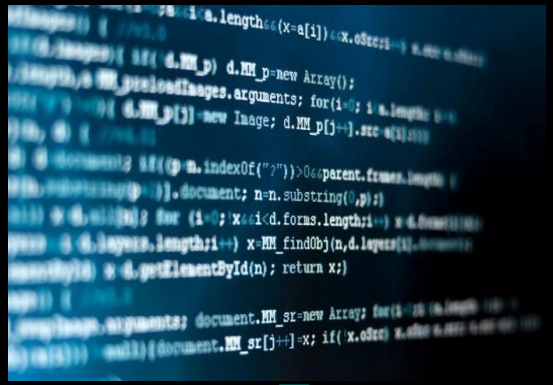


Ciclo de vida de un programa



Ensamblador

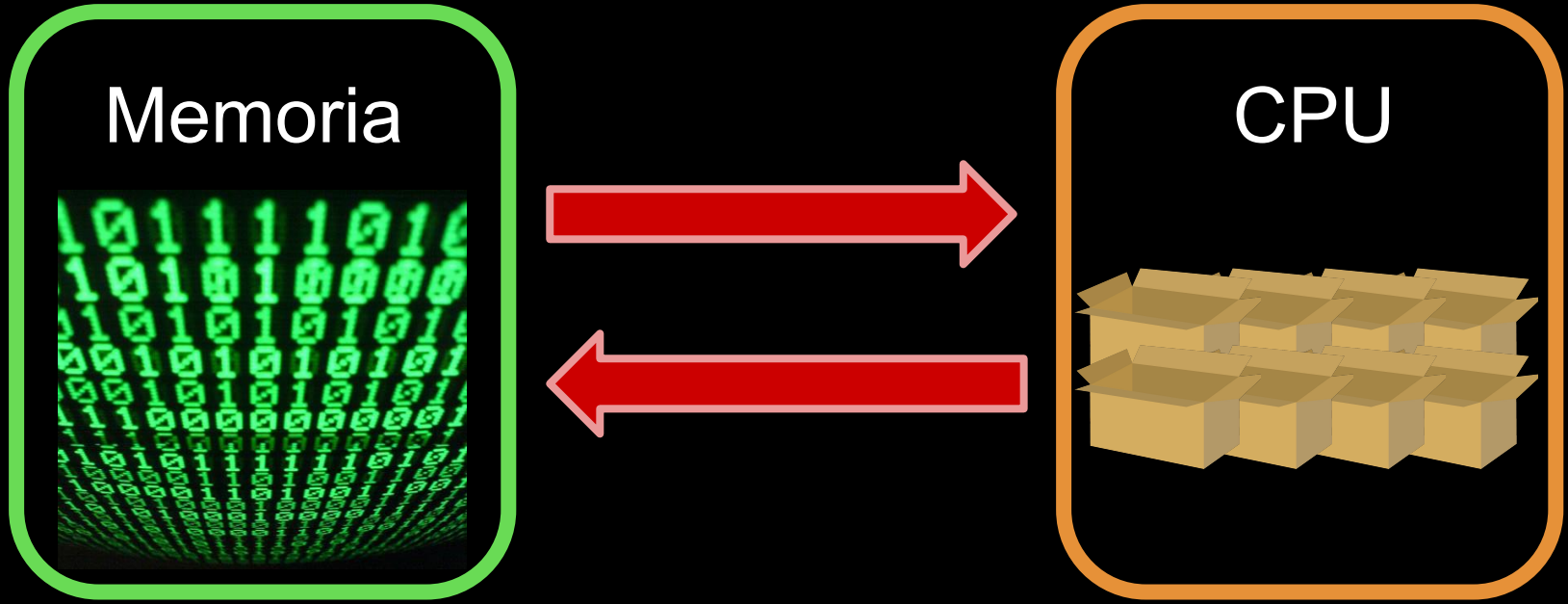
Ciclo de vida de un programa



Ensamblador

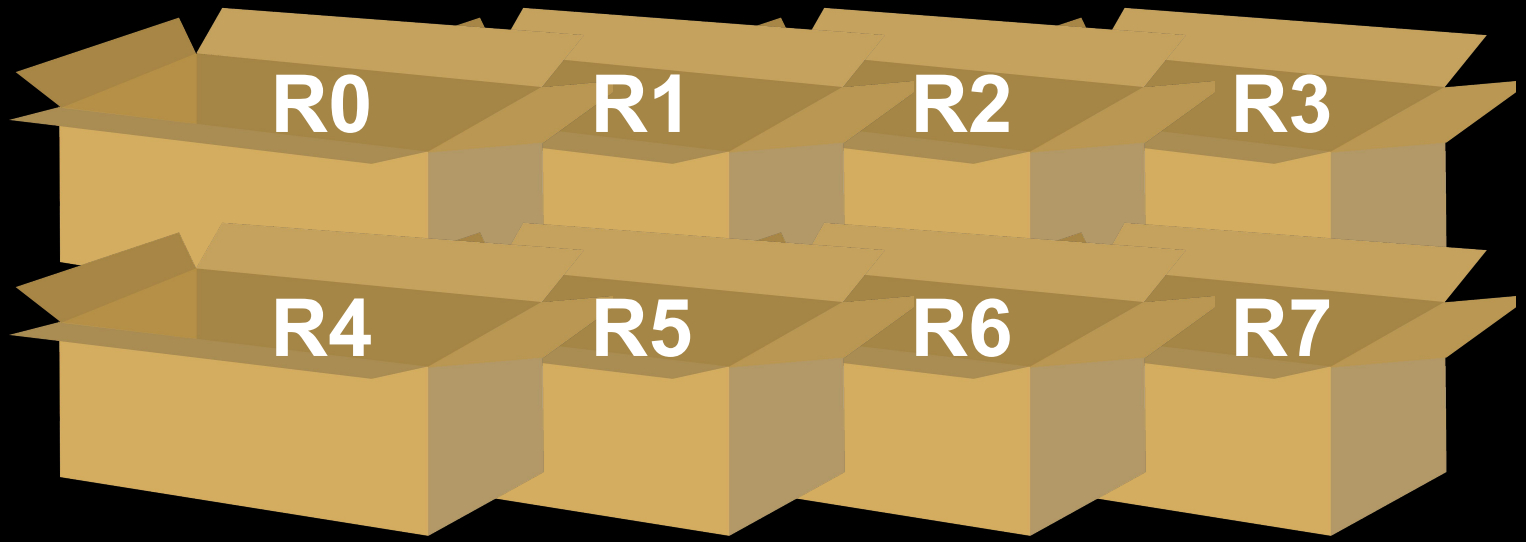
Ciclo de vida de un programa

Ciclo de vida de un programa



Arquitecturas Q: Q1

Arquitecturas Q: Q1



Arquitecturas Q: Q1

Operandos

Arquitecturas Q: Q1

Operandos

- Registro

Arquitecturas Q: Q1

Operandos

- Registro
- Constante

Arquitecturas Q: Q1

Instrucciones

Operación	Código	Efecto
MUL	0000	Dest <- Dest * Origen
MOV	0001	Dest <- Origen
ADD	0010	Dest <- Dest + Origen
SUB	0011	Dest <- Dest - Origen
DIV	0111	Dest <- Dest % Origen

Arquitecturas Q: Q1

Modos de direccionamiento

- Inmediato: el operando está en la instrucción
- Registro: el operando es un registro

Arquitecturas Q: Q1

Ejemplos:

- MOV R7, R1
- MUL R5, 5
- DIV R3, 10

Arquitecturas Q: Q1

Instrucciones:

MUL, MOV, ADD, SUB, DIV

Arquitecturas Q: Q1

Instrucciones:

MUL, MOV, ADD, SUB, DIV

Operandos (Modos de direccionamiento):

Arquitecturas Q: Q1

Instrucciones:

MUL, MOV, ADD, SUB, DIV

Operandos (Modos de direccionamiento):

Registro (modo registro)

Arquitecturas Q: Q1

Instrucciones:

MUL, MOV, ADD, SUB, DIV

Operandos (Modos de direccionamiento):

Registro (modo registro)

Constante (modo inmediato)

Arquitecturas Q: Q1

Operación	Código
MUL	0000
MOV	0001
ADD	0010
SUB	0011
DIV	0111

Modo	Código
Registro	100RRR
Inmediato	000000

Cod Op (4 bits)	**Modo destino** (6 bits)	**Modo origen** (6 bits)	**Origen** (16 bits)

Arquitecturas Q: Q1

Operación	Código
MUL	0000
MOV	0001
ADD	0010
SUB	0011
DIV	0111

Modo	Código
Registro	100RRR
Inmediato	000000

Cod Op (4 bits)	**Modo destino** (6 bits)	**Modo origen** (6 bits)	**Origen** (16 bits)

Ensamblar: MOV R3, 0xFFEF; ADD R5, R3

Arquitecturas Q: Q1

Operación	Código
MUL	0000
MOV	0001
ADD	0010
SUB	0011
DIV	0111

Modo	Código
Registro	100RRR
Inmediato	000000

Cod Op (4 bits)	**Modo destino** (6 bits)	**Modo origen** (6 bits)	**Origen** (16 bits)

Ensamblar: MOV R3, 0xFFEF; ADD R5, R3

Desensamblar: 0000 1000 0110 0000
0010 1000 0000 0000
0000 0000 0000 0111