INSTRUCCIONES CONDICIONALES

Iniciar

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

Objetivos

Los objetivos que se pretenden alcanzar en este recurso son los siguientes:

- Aprender cuáles son los mecanismos de ejecución de código no secuencial.
- Definir las instrucciones para modificar la secuencia de ejecución.
- Identificar las sentencias condicionales en ensamblador.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Introducción

A lo largo de este recurso se verá la importancia de las instrucciones condicionales, que forman la base para resolver (computacionalmente hablando) cualquier tipo de problema y sin las cuales sería prácticamente imposible. Se hará hincapié en cada una de las instrucciones condicionales y se verá mediante un ejemplo su funcionamiento.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1

2

3

4

5

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1
2
3
4
5

Ejecutar código condicional, dependiendo de una condición previa:

- Ejemplo: instrucciones if/then/else, switch/case, etc.

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1
2
3
4
5

Repetir algunas partes del código (bucles):

- Ejemplo: for(), while(), do…while(), etc.

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1
2
3
4
5

Saltar a (o volver a) rutinas y funciones, para permitir reutilización de código y niveles de abstracción:

- Ejemplo: CALL rutina, RET (de rutina).

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1
2
3
4
5

Asistir a peticiones de

interrupción de dispositivos.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

La mayoría de las CPU ejecutan el código principalmente de una manera secuencial (una instrucción después de otra). Sin embargo, las CPU también deben permitir alterar esta secuencia ordenada con el fin de:

1
2
3
4
5

Saltar entre hilos de un proceso para permitir multitarea.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Contador

Fase de carga

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Contador

Fase de carga

El contador de programa (PC) es un registro que almacena la dirección de memoria de la siguiente instrucción a ejecutar:

- Atmega328p: registro de 16 bits. Memoria máxima 64K (aunque solo tiene 32K).

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

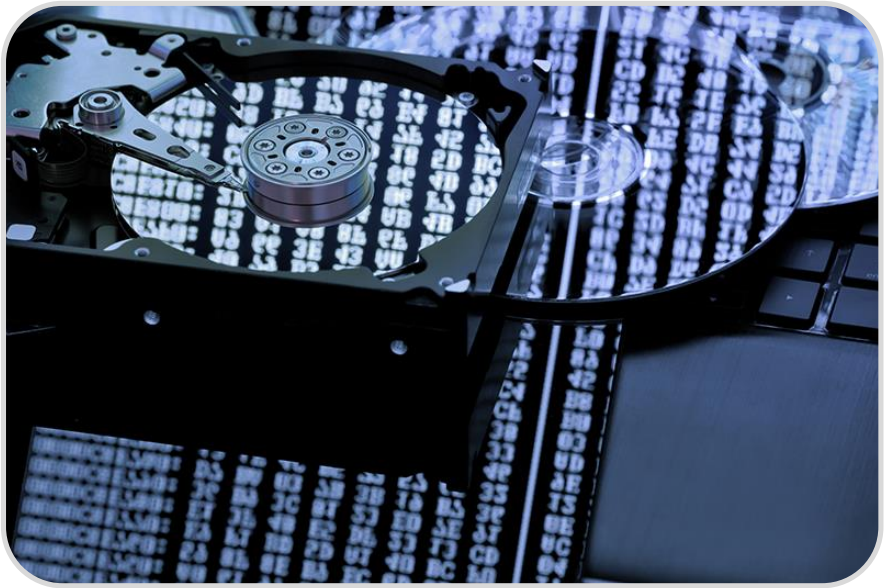
Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Contador

Fase de carga



En la fase de carga de instrucción se trae de la memoria del programa la dirección indicada por el PC:

- La instrucción se almacena en el registro IR, para ser decodificada y ejecutada.

Mas info

Anterior

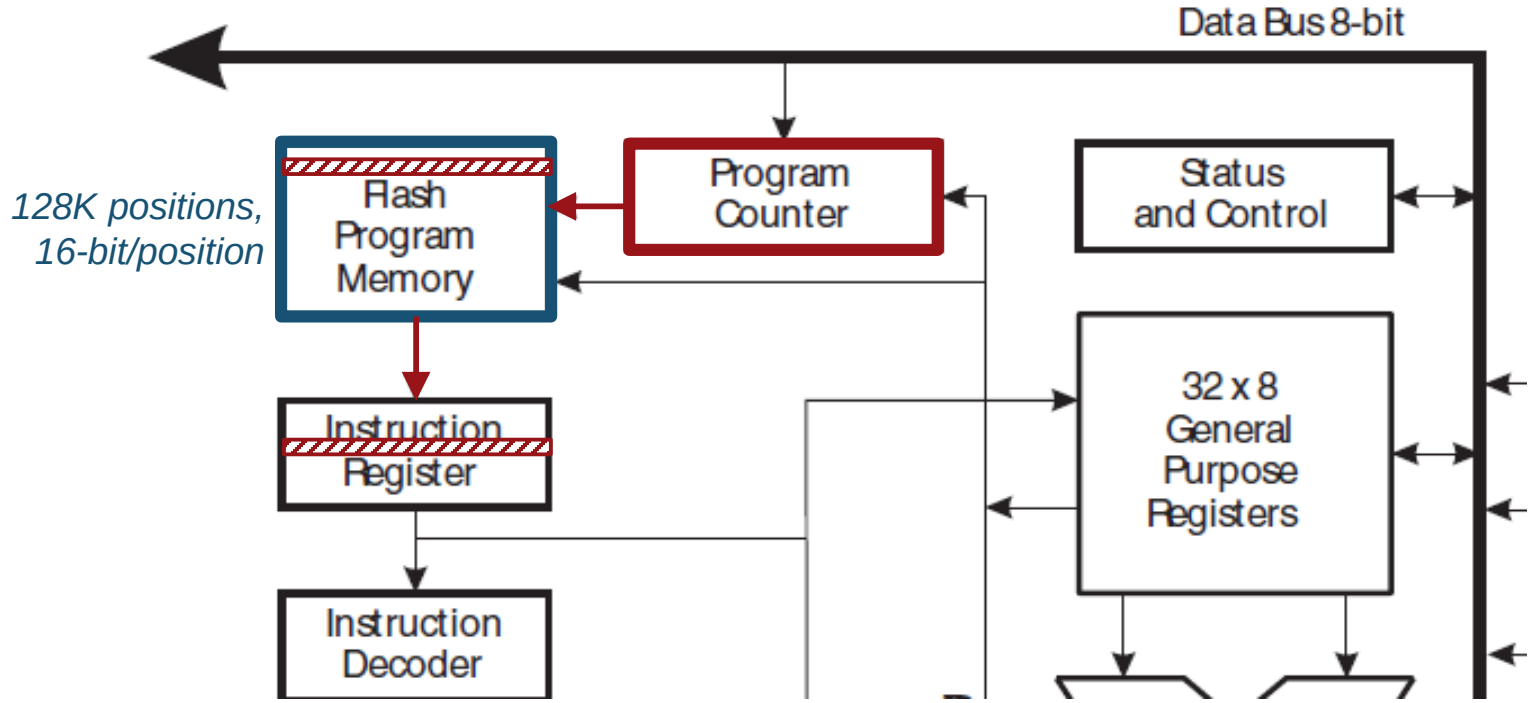
© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Mecanismo de ejecución de código no secuencial



© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Poscarga

Instrucción de salto

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

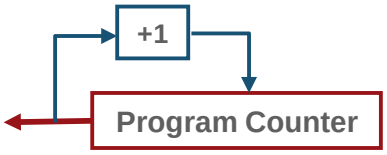
Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Poscarga

Instrucción de salto

Después de cada carga de instrucción, el PC se incrementa en 1, quedando preparado para cargar la siguiente instrucción.



Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Mecanismo de ejecución de código no secuencial

Poscarga

Instrucción de salto

Si la instrucción es una instrucción de salto, entonces el registro PC se sobrescribe con la dirección de salto, lo que permite alterar la secuencia secuencial de ejecución.

Mas info

Anterior

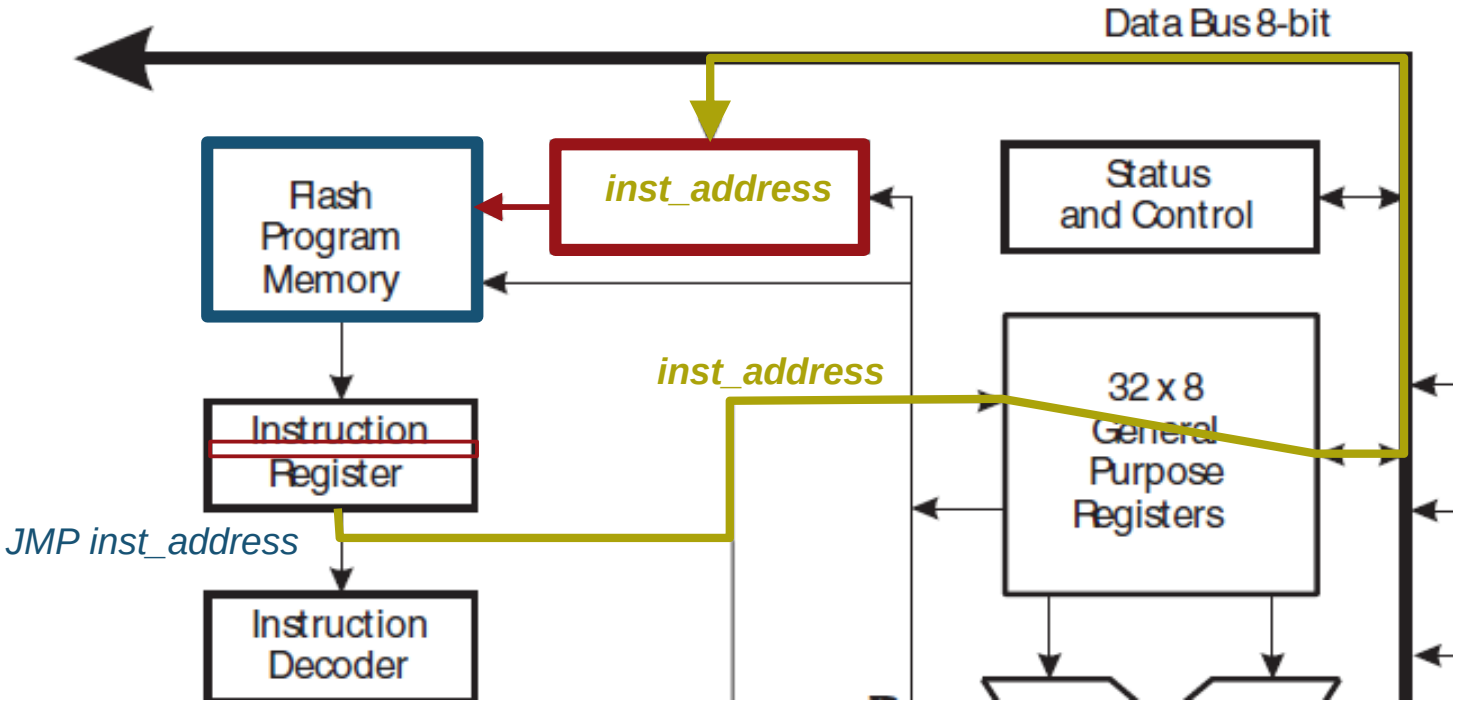
© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Mecanismo de ejecución de código no secuencial



© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

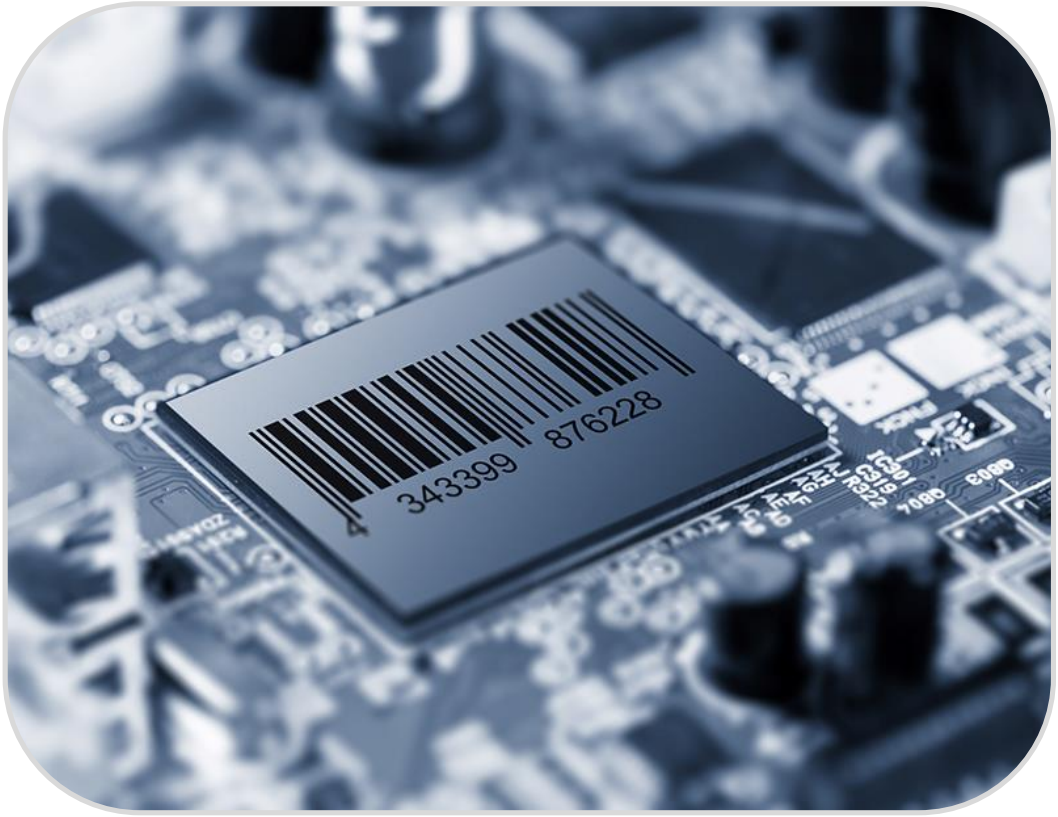
Haz clic sobre las zonas marcadas para ampliar la información

Registro de instrucciones

Almacena la instrucción

La longitud del “OpCode” puede variar

Completar el código



© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Registro de instrucciones

Almacena la instrucción

La longitud del "OpCode" puede variar

Completar el código

Almacena la instrucción en ejecución para:

- Extraer el código de operación y enviarlo al decodificador de instrucciones.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Registro de instrucciones

Almacena la instrucción

La longitud del "OpCode" puede variar

Completar el código

En los procesadores Atmel, es al menos de 4 bits (del B15-B12), y se conoce como código de operación "inicial".

0101	$k_7k_6k_5k_4$	$d_3d_2d_1d_0$	$k_3k_2k_1k_0$
0000	$11r_4d_4$	$d_3d_2d_1d_0$	$r_3r_2r_1r_0$
1001	$001r_4$	$r_3r_2r_1r_0$	0000
1110	1111	$d_3d_2d_1d_0$	1111
1001	0101	1000	1000

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Registro de instrucciones

Almacena la instrucción

La longitud del “OpCode” puede variar

Completar el código

Analizando este código de operación "inicial", el decodificador puede determinar si hay más bits para completar el código de operación "extendida“:

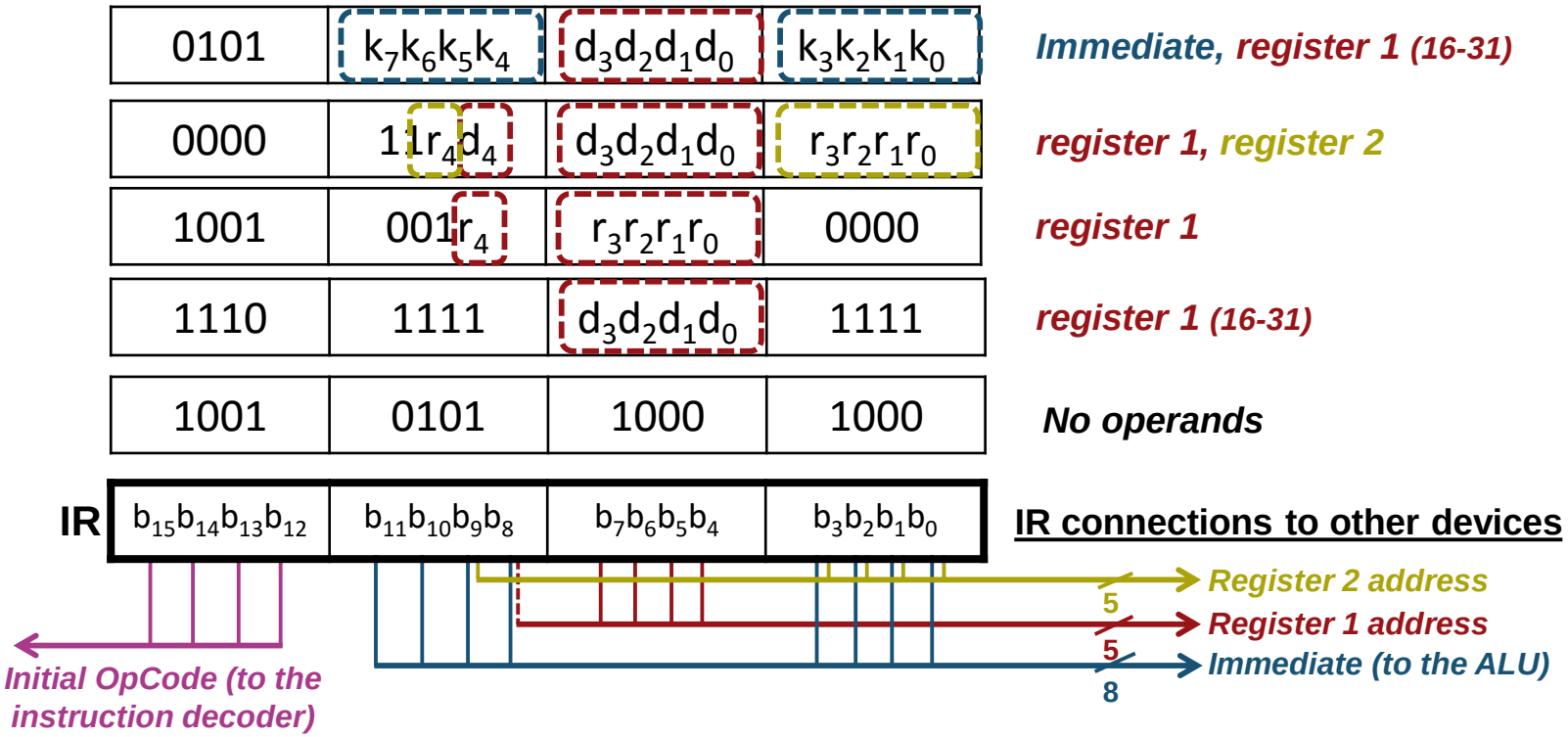
- Una vez que se extrae el código de operación "extendida", el decodificador de instrucciones desencadena los circuitos específicos (unidad de control) que ejecuta la instrucción.

© Universidad Europea de Madrid. Todos los derechos reservados.

Registro de instrucciones: operandos

Una vez determinado el código de operación, se pueden determinar los operandos:

- El número de operandos varían para cada instrucción, sin embargo, su variedad es limitada, y se colocan siempre en las mismas posiciones:



Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional

La instrucción más básica para modificar la secuencia de ejecución es la instrucción de salto incondicional. Esta instrucción, provoca un cambio en la secuencia de ejecución, sí o sí.

JMP (direct JuMP)

1001	$010a_{21}$	$a_{20}a_{19}a_{18}a_{17}$	$110a_{16}$
$a_{15}a_{14}a_{13}a_{12}$	$a_{11}a_{10}a_9a_8$	$a_7a_6a_5a_4$	$a_3a_2a_1a_0$

Formato

Características

Dirección

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional

La instrucción más básica para modificar la secuencia de ejecución es la instrucción de salto incondicional. Esta instrucción, provoca un cambio en la secuencia de ejecución, si o si.

JMP (direct JuMP)

1001	$010a_{21}$	$a_{20}a_{19}a_{18}a_{17}$	$110a_{16}$
$a_{15}a_{14}a_{13}a_{12}$	$a_{11}a_{10}a_9a_8$	$a_7a_6a_5a_4$	$a_3a_2a_1a_0$

Formato

Características

Dirección

- Formato: JMP a.
- Carga la dirección absoluta de salto en el contador de programa (PC←a), donde 0 < a < 4M.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional

La instrucción más básica para modificar la secuencia de ejecución es la instrucción de salto incondicional. Esta instrucción, provoca un cambio en la secuencia de ejecución, si o si.

JMP (direct JuMP)

1001	$010a_{21}$	$a_{20}a_{19}a_{18}a_{17}$	$110a_{16}$
$a_{15}a_{14}a_{13}a_{12}$	$a_{11}a_{10}a_9a_8$	$a_7a_6a_5a_4$	$a_3a_2a_1a_0$

Formato

Características

Dirección

Es una instrucción grande y lenta:

- Ocupa dos posiciones de memoria de programa.
- Consume tres ciclos de reloj en ejecutarse.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional

La instrucción más básica para modificar la secuencia de ejecución es la instrucción de salto incondicional. Esta instrucción, provoca un cambio en la secuencia de ejecución, si o si.

JMP (direct JuMP)

1001	$010a_{21}$	$a_{20}a_{19}a_{18}a_{17}$	$110a_{16}$
$a_{15}a_{14}a_{13}a_{12}$	$a_{11}a_{10}a_9a_8$	$a_7a_6a_5a_4$	$a_3a_2a_1a_0$

Formato

Características

Dirección

La dirección de destino puede ser indicada mediante una etiqueta en el código.
Ejemplo:

```
instructions…
         JMP my_first_label
         ……
my_first_label:  instructions  ;my_first_label ≡ program address
                 instructions
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional corto

Una instrucción más rápida (y más pequeña) para hacer un salto incondicional es:

RJMP (Relative JuMP)

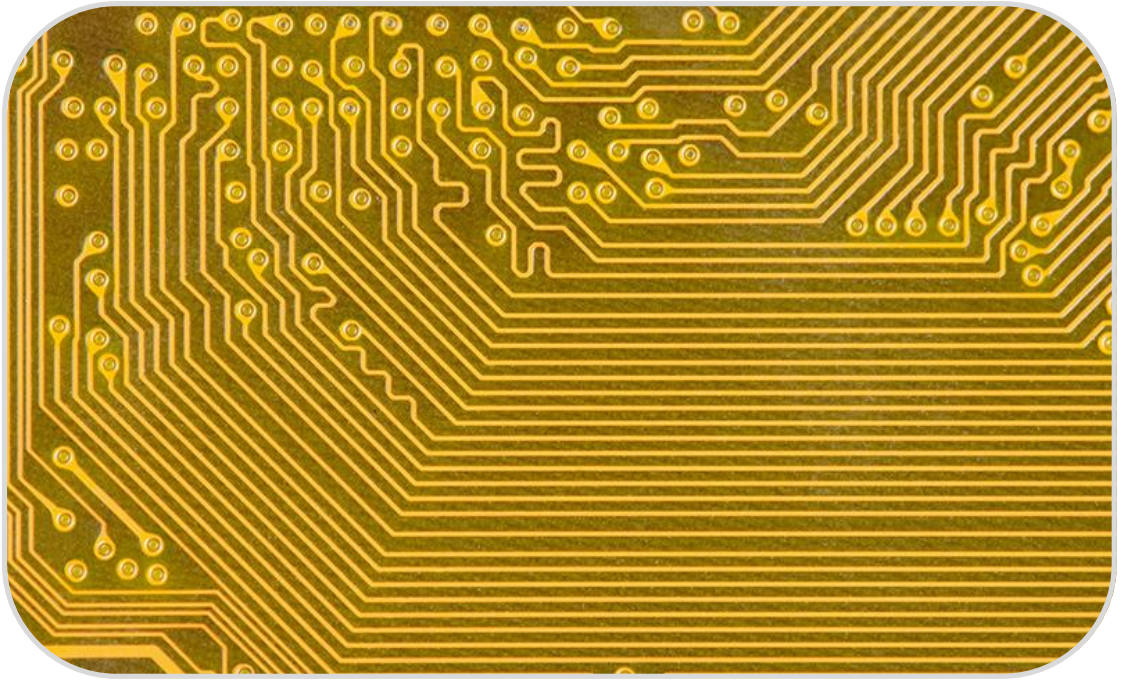
1100	$r_{11}r_{10}r_9r_8$	$r_7r_6r_5r_4$	$r_3r_2r_1r_0$

Formato

Salto relativo

Diferencia de JMP

Dirección



© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional corto

Una instrucción más rápido (y más pequeño) para hacer un salto incondicional es:

RJMP (Relative JuMP)

1100	$r_{11}r_{10}r_9r_8$	$r_7r_6r_5r_4$	$r_3r_2r_1r_0$

Formato

Salto relativo

Diferencia de JMP

Dirección

Formato: RJMP r

- Incrementa las posiciones de contador del programa en “r”.
- PC ← PC + r + 1, donde “r” puede ser positivo o negativo, por lo que -2K < r < 2K.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional corto

Una instrucción más rápido (y más pequeño) para hacer un salto incondicional es:

RJMP (Relative JuMP)

1100	$r_{11}r_{10}r_9r_8$	$r_7r_6r_5r_4$	$r_3r_2r_1r_0$

- Formato
- Salto relativo
- Diferencia de JMP
- Dirección

Incrementa o decrementa (si r es negativo) el valor actual de PC. Por eso es un salto relativo a la posición actual.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional corto

Una instrucción más rápido (y más pequeño) para hacer un salto incondicional es:

RJMP (Relative JuMP)

1100	$r_{11}r_{10}r_9r_8$	$r_7r_6r_5r_4$	$r_3r_2r_1r_0$

Formato

Salto relativo

Diferencia de JMP

Dirección

Este salto es más corto y más rápido que JMP:

- Ocupa una posición de memoria de programa.
- Consume dos ciclos de reloj en ejecutarse.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucción de salto incondicional corto

Una instrucción más rápido (y más pequeño) para hacer un salto incondicional es:

RJMP (Relative JuMP)

1100	$r_{11}r_{10}r_9r_8$	$r_7r_6r_5r_4$	$r_3r_2r_1r_0$

Formato

Salto relativo

Diferencia de JMP

Dirección

La dirección de destino puede ser indicada mediante una etiqueta en el código:

```
instructions…
RJMP another_label
……                  ;distance r ≡ +7 program mem. positions
……
another_label:       ;another_label ≡ +7 (distance from RJMP to here)
         instructions…
```

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

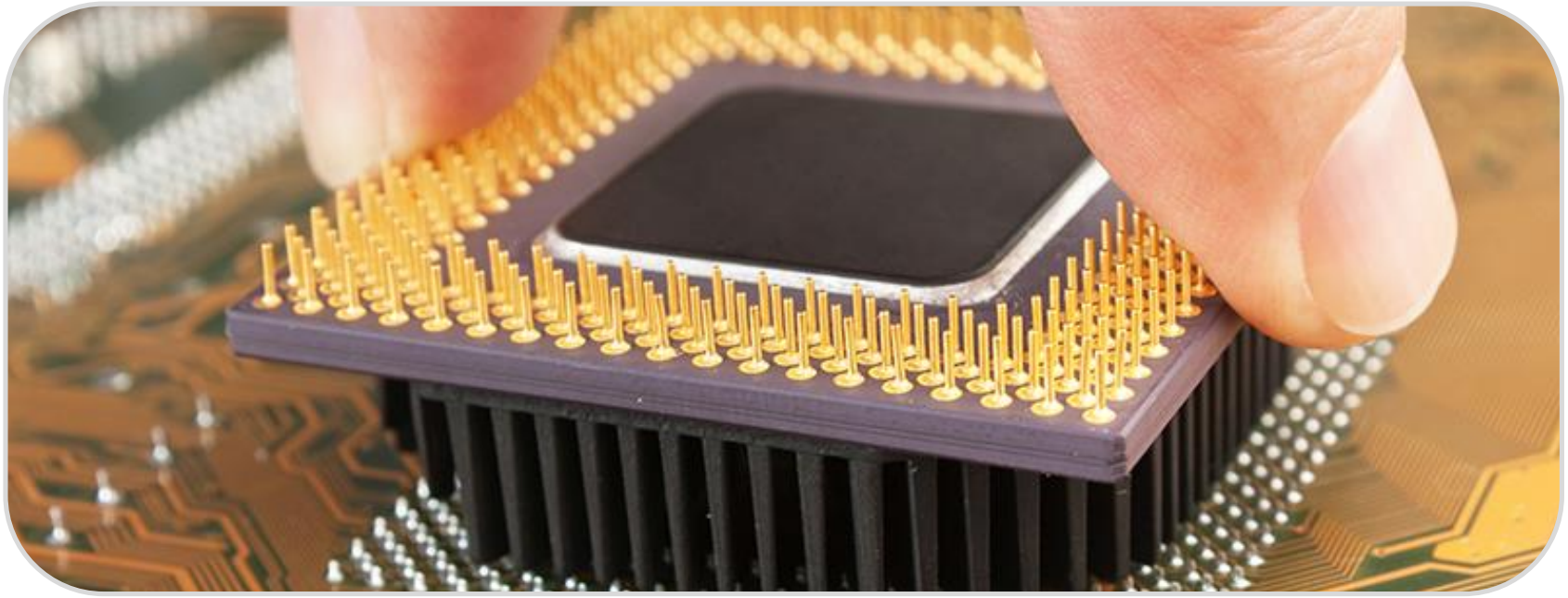
Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Ejemplo: bucle infinito principal “main loop”

Ejemplo de programa mínimo



Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Ejemplo: bucle infinito principal “main loop”

Ejemplo de programa mínimo

```
1.  INCLUDE "m328pdef.inc"          ;has pin and port "human" names for ATMega328p.
2.   .device ATmega328p             ;check for invalid instructions in ATMega328p  processor
3.
4.   .ORG 0x00000                   ;Program memory position 0x00000 is executed after each reset
5.          RJMP main               ;reset “interrupt vector” (address to jump to after a reset)
6.
7.   .ORG 0x00072                   ;leave room for other “interrupt vectors”
8.    main:     LDI R16,0x07        ;do something
9.          ……..                    ;do something
10.         ……..
11.         RJMP main               ;provoke an infinite loop
12.                                 ;JMP main ;use this option if the relative distance to “main” is bigger than 2K
13.
14.  .EXIT
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones de salto condicional en igualdad

Los saltos condicionales deben seguir una operación que afecta a los flag (por lo general una comparación, CP). Las principales instrucciones de este tipos son:

BREQ (BRanch if EQual)

BRNE (BRanch if NotEqual)

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones de salto condicional en igualdad

Los saltos condicionales deben seguir una operación que afecta a los flag (por lo general una comparación, CP). Las principales instrucciones de este tipos son:

BREQ (BRanch if EQual) | BRNE (BRanch if NotEqual)

- Formato: BREQ r

1111	$00r_6r_5$	$r_4r_3r_2r_1$	r_0001

  Si el flag Z=1 (comparación en igualdad o resultado de cero) modifica el contador de programa (PC) en r posiciones (PC←PC+r+1).

  En otro caso no hace nada, manteniendo el valor de PC en (PC←PC+1):

  - r puede ser positivo o negativo, con -64 < r < 63.
  - Modifica el valor de PC, por lo que se considera un salto relativo.

- La dirección de destino puede ser indicada mediante una etiqueta en el código.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones de salto condicional en igualdad

Los saltos condicionales deben seguir una operación que afecta a los flag (por lo general una comparación, CP). Las principales instrucciones de este tipos son:

BREQ (BRanch if EQual) | BRNE (BRanch if NotEqual)

- Formato: BRNE r

1111	$01r_6r_5$	$r_4r_3r_2r_1$	r_0001

Si el flag Z=0 (comparación en desigualdad o resultado distinto de cero) modifica el contador de programa (PC) en r posiciones (PC←PC+r+1).

En otro caso no hace nada, manteniendo el valor de PC en (PC←PC+1):

  - r puede ser positivo o negativo, con -64 < r < 63.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Ejemplo de Instrucciones de salto condicional en igualdad

Ejemplo de retardo básico

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Ejemplo de Instrucciones de salto condicional en igualdad

Ejemplo de retardo básico

```
.DSEG                   ;indicate we are going to use data memory addresses

.CSEG
main:
        ...             ;do something
        LDI R16, 10     ;same as LDI R16 at 10  ;init R16 to the max value
contador:
        DEC R16         ;decrement R16 in one. This already affects Z flag (no CP needed)
        BRNE contador   ;if the result wasn't zero (so Z=0) jump back to delay
        ...             ;do something
.EXIT
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Otras instrucciones de salto condicional

Cada instrucción de salto, comprueba varios flag del registro SREG, según estas comprobaciones, la variantes de instrucciones de salto son:

flag	Salta si flag=0	Salta si flag=1	Descripción	valido con valores con/sin signo
C	BRCC	BRCS	BRanch if Carry is Cleared (0) / if Carry is Set (1)	Sin signo
C	BRSH	BRLO	BRanch if Same or Higher (C=0) / if LOwer (C=1)	Sin signo
Z	BRNE	BREQ	BRanch if Z=0 (not 0, Not Equal) / if Z=1 (0, EQual)	ambos
N	BRPL	BRMI	BRanch if PLus (N=0) / if MInus (N=1)	Con signo
V	BRVC	BRVS	BRanch if oVerflow is Cleared (0) / if oVreflow is Set (1)	ambos
S	BRGE	BRLT	BRanch if Greater or Equal (S=0) / if Less Than (S=1)	Con signo
H	BRHC	BRHS	BRanch if H is Cleared (0) / if H is Set (1)	-
T	BRTC	BRTS	BRanch if bit-memory T is Cleared (0) / if T is Set (1)	-
I	BRID	BRIE	BRanch if Interrupts Disabled (0) / if Interrupts Enabled (1)	-

Mas info

Anterior

Siguiente

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Otras instrucciones de salto condicional

Volver

También se pueden hacer saltos, preguntando por un bit en concreto del registro SREG directamente:

- BRBC n, r : salta a r si bit n en el registro SREG esta a cero.
- BRBS n, r : salta a r si bit n en el registro SREG esta a uno.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Instrucciones de “saltar una”

Conocidas como “Next instruction skipping”, estas instrucciones saltan la instrucción siguiente, si la condición es verdadera. En otro caso, ejecutan la instrucción siguiente normalmente:

- Si la condición es verdadera, se incrementa el contador del programa en una posición, provocando que la siguiente instrucción no se ejecute.
- Si la condición no se cumple, no hace nada, permitiendo que se ejecute la secuencia estándar de código secuencial.

Mas info

Anterior

© Universidad Europea de Madrid. Todos los derechos reservados.

Siguiente

Instrucciones condicionales

Haz clic sobre la zona marcada para ampliar la información

Instrucciones de “saltar una”

Volver

En la siguiente tabla se muestran las principales instrucciones de este tipo:

Instrucción	Formato	Descripción
CPSE	CPSE Rd, Rr	ComPare and Skip if Equal: si Rd=Rr, saltar siguiente instrucción
SBRC	SBRC Rr, b	Skip If Bit b in Register Rr is Cleared: si bit b en Rr es 0, saltar siguiente instrucción
SBRS	SBRS Rr, b	Skip If Bit b in Register Rr is Set: si bit b en Rr es 1, saltar siguiente instrucción
SBIC	SBIC P, b	Skip if Bit b in I/o port is Cleared: sif bit b en puerto P es 0, saltar siguiente instrucción
SBIS	SBIS P, b	Skip if Bit b in I/o port is Set: if bit b in port P is 1, saltar siguiente instrucción

Ejemplo:

```
SBIS PORTB, 7     ;comprueba si el PORTB, bit 7 es 1.
RJMP bit_es_0     ;instrucción saltada si la condición anterior se cumple.
……                ;siguiente instrucción ejecutada si la condición se cumple.
```

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “for( )”

Todos los tipos de bucles deben ser implementados utilizando comparaciones y saltos condicionales. No existe instrucciones tipo for, while, etc.

Versión en C

Procesadores Atmel

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “for( )”

Todos los tipos de bucles deben ser implementados utilizando comparaciones y saltos condicionales. No existe instrucciones tipo for, while, etc.

Versión en C | Procesadores Atmel

El bucle más básico es el bucle for(). Se tomará la versión en C como ejemplo:

```
1. for (contador = 0; contador <10; contador ++) {...}
```

- Se inicializa el contador a 0, y el bucle se repetirá mientras que el contador es <10; después de cada iteración, el contador se incrementa en 1.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “for( )”

Todos los tipos de bucles deben ser implementados utilizando comparaciones y saltos condicionales. No existe instrucciones tipo for, while, etc.

Versión en C | Procesadores Atmel

Esta instrucción para los procesadores Atmel se implementa:

```
CLR R16                 ;use R16 as counter
Repeat: …
        …               ;the code to repeat goes here
        …
        INC R16         ;increase the counter
        CPI R16, 10     ;compare with the number of iterations to do
        BRNE Repeat     ;jump if not equal. Also BRLO Repeat (jump if lower) can be used
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “do ….. while( )”

Versión en C

Procesadores Atmel

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “do ….. while( )”

Versión en C | Procesadores Atmel

Una vez mas, se tomará C como lenguaje de ejemplo para esta instrucción:

```
1.  do {……} while(condición)
```

- El bucle se ejecuta incondicionalmente, una vez terminada una iteración, se comprueba la condición de fin, si se cumple, el bucle itera una vez mas. Y así, hasta que la condición se haga falsa.
- Es importante observar, que aunque la condición sea falsa desde el principio, este bucle se ejecuta al menos una vez.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “do ….. while( )”

Versión en C | Procesadores Atmel

Esta instrucción para los procesadores Atmel se implementa:

```
Repeat: …
        …               ;the code to repeat goes here
        …
        CP R0, R1       ;check the condition (this is just an example)
        BREQ Repeat     ;jump back if the condition is met
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “while( )”

Versión en C

Procesadores Atmel

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “while( )”

Versión en C | Procesadores Atmel

Una vez mas, se tomará C como lenguaje de ejemplo para esta instrucción:

```
1. while(condición){…}
```

- Se comprueba la condición del bucle, si la condición es verdadera, el bucle itera una vez más. Y así, hasta que la condición se haga falsa.
- Es importante observar, que estando la condición en falsa inicialmente, el bucle no itera ninguna vez.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Bucles: instrucción de alto nivel “while( )”

Versión en C | Procesadores Atmel

Esta instrucción para los procesadores Atmel se implementa:

```
Repeat:
        CP R0, R1               ;firstly check the condition (this is just an example)
        BRNE end_loop           ;if the condition is not met, the loop is finished
        …
        …           ;the code to repeat goes here
        …
        RJMP Repeat             ;try to repeat the loop
end_loop:                       ;continue after the loop
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…}”

Versión en C

Procesadores Atmel

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…}”

Versión en C | Procesadores Atmel

Una vez mas, se tomará C como lenguaje de ejemplo para esta instrucción:

```
1. if(condición){…}
```

- Comprueba el estado de la condición. Si es verdadera, ejecutar el código entre paréntesis. De lo contrario, lo salta y continua la ejecución después de la última instrucción del bloque de paréntesis.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…}”

Versión en C | Procesadores Atmel

Esta instrucción para los procesadores Atmel se implementa:

```
CP R0, R1               ;firstly check the condition (this is just an example)
        BRNE continue1  ;if the condition is not met, ignore the conditional code and continue
        …
        …     ;the conditional code goes here
        …
continue:               ;continue the normal execution
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…} else {…}”

Versión en C

Procesadores Atmel

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…} else {…}”

Versión en C | Procesadores Atmel

Una vez mas, se tomará C como lenguaje de ejemplo para esta instrucción:

```
1. if(condición){...} else {...}
```

- Comprueba el estado de la condición. Si es verdad, ejecuta el bloque “then". De lo contrario, ejecuta el bloque “else". Bien ya sea ejecutando un bloque o el otro, continua con la ejecución normal del código.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Instrucciones condicionales: sentencia “If() then {…} else {…}”

Versión en C | Procesadores Atmel

Esta instrucción para los procesadores Atmel se implementa:

```
            CP R0, R1   ;firstly check the condition (this is just an example)
            BRNE else_code  ;if the condition is not met, jump to the “else” code
            …
            …  ;the “then” conditional code goes here
            …
            RJMP continue  ;the “then” code has been executed, now jump over the “else” code
else_code:  …
            …  ;the “else” conditional code goes here
            …
continue:   …  ;continue the normal execution
```


© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Sentencias condicionales compuestas: “if((condit1)AND/OR(condit2))…”

Las condiciones compuestas deben ser analizadas una por una, como sentencias individuales:

if((condition1)AND(condition2))

if((condition1)OR(condition2))

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Sentencias condicionales compuestas: “if((condit1)AND/OR(condit2))…”

Las condiciones compuestas deben ser analizadas una por una, como sentencias individuales:

if((condition1)AND(condition2))

if((condition1)OR(condition2))

```
1. if((condition1)AND(condition2))
```

Se comprueba la 1º condición:

- Si es falsa, dado que esto provoca que toda la condición sea falsa, saltamos directamente a la parte “else”.
- Si es verdadera, entonces se comprueba la 2º condición:
  - Si esta es verdadera, se ejecuta la parte “then”, sino la parte “else”.

© Universidad Europea de Madrid. Todos los derechos reservados.

Instrucciones condicionales

Haz clic sobre las zonas marcadas para ampliar la información

Sentencias condicionales compuestas: “if((condit1)AND/OR(condit2))…”

Las condiciones compuestas deben ser analizadas una por una, como sentencias individuales:

if((condition1)AND(condition2))

if((condition1)OR(condition2))

```
1. if((condition1)OR(condition2))
```

Se comprueba la 1º condición:

- Si es verdadera, dado que esto provoca que toda la condición sea verdadera, saltamos directamente a la parte “then”.
- Si es falsa, entonces se comprueba la 2º condición:
  - Si esta es verdadera, se ejecuta la parte “then”, sino la parte “else”.

© Universidad Europea de Madrid. Todos los derechos reservados.

Universidad Europea

LAUREATE INTERNATIONAL UNIVERSITIES

PDF contenidos