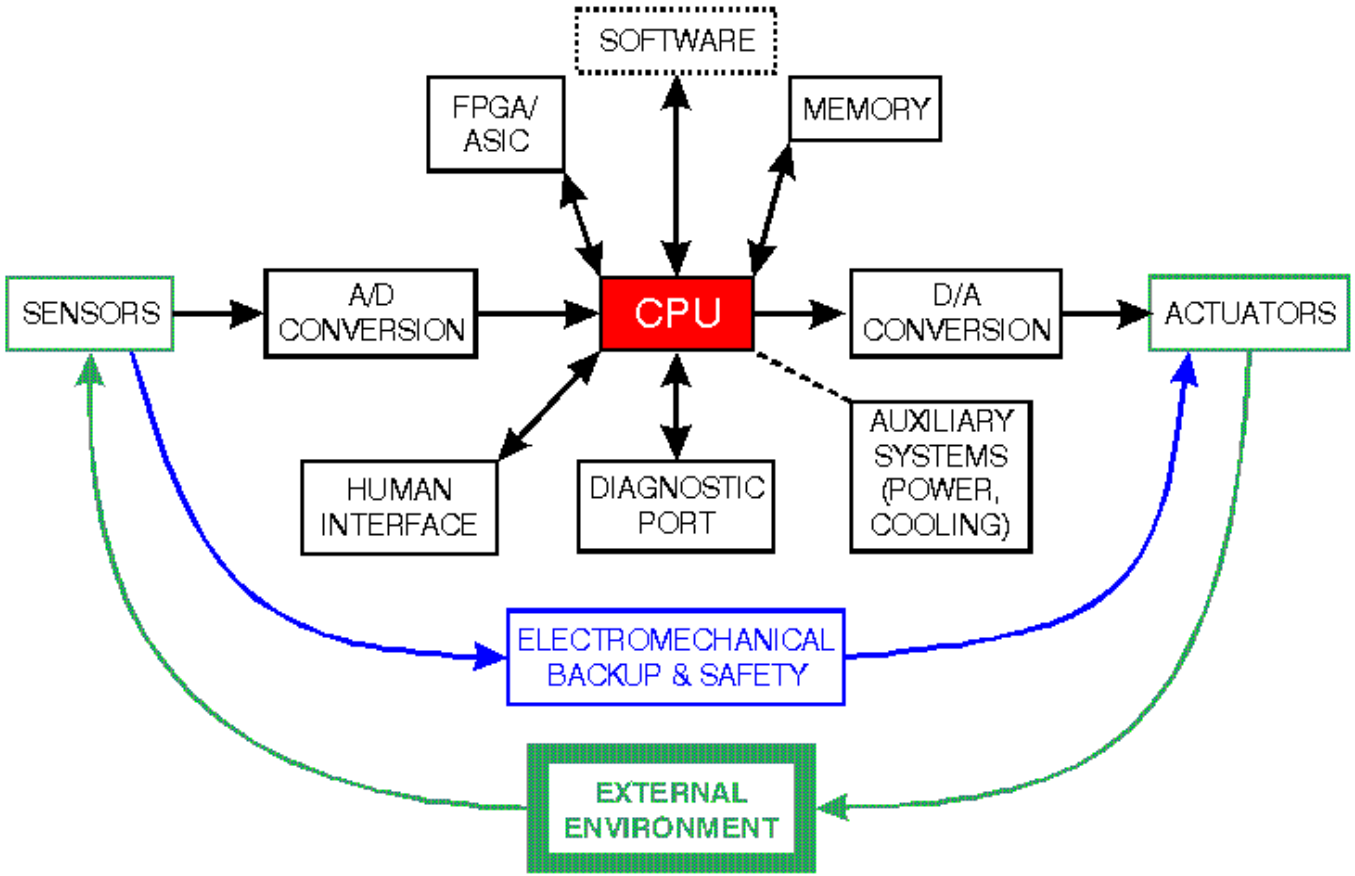
Programación de Sistemas

Mtro. en IA José Rafael Rojano Cáceres

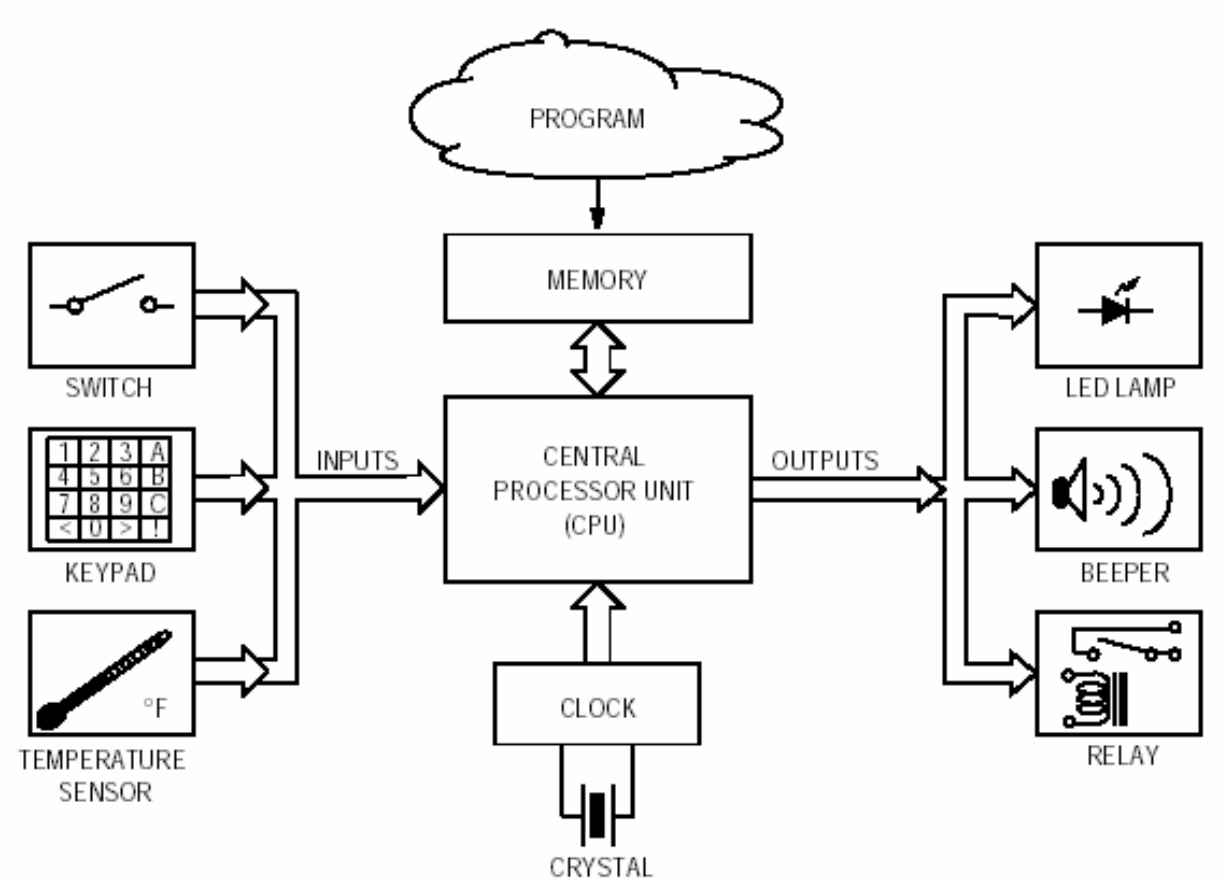
rrojano@gmail.com

http://www.uv.mx/rrojano

Elementos de un sistema embebido

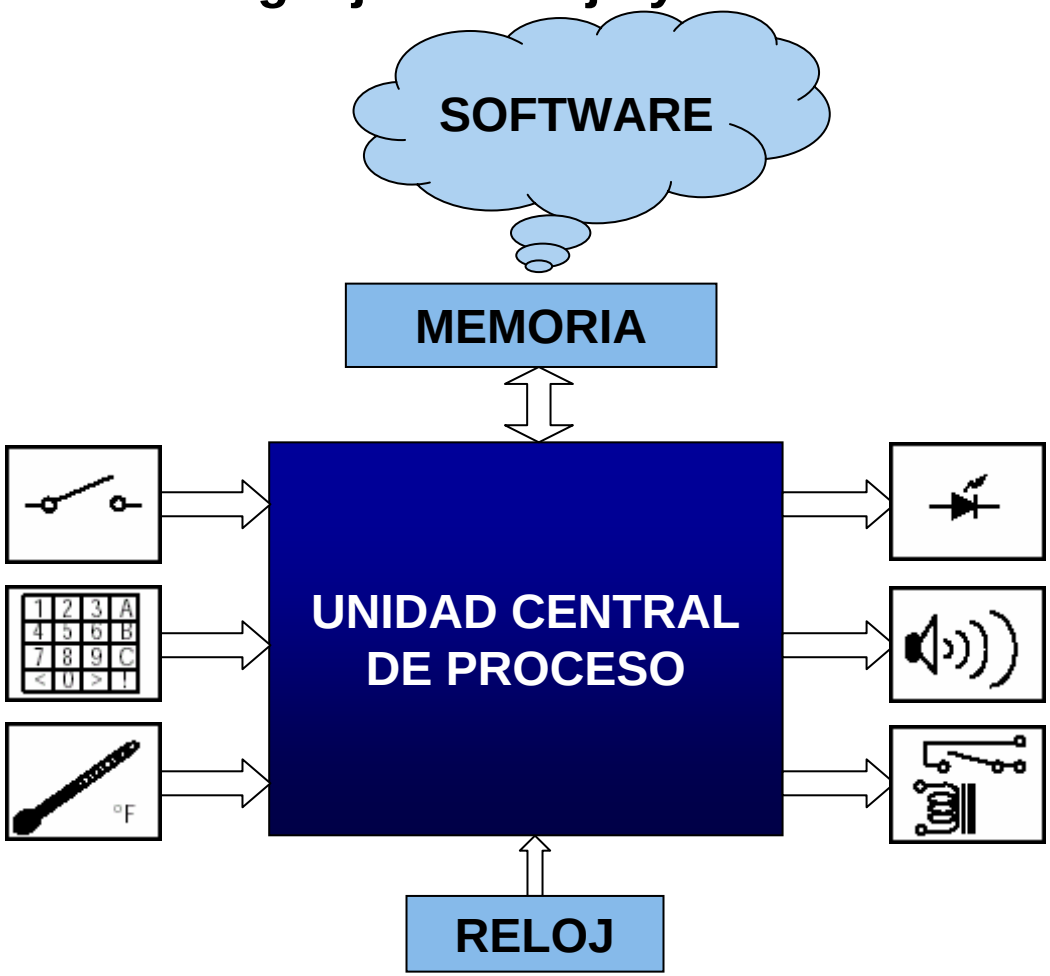


Elementos de un sistema embebido



Desarrollo de Firmware

Lenguajes de Bajo y Alto Nivel.



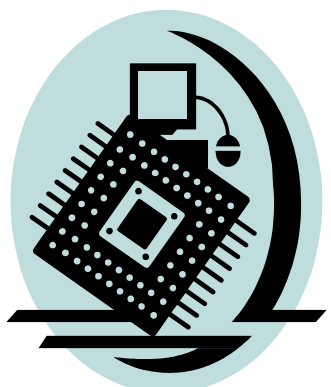
Desarrollo de Firmware

Sumar dos valores y comparar si el resultado es mayor o igual a 10

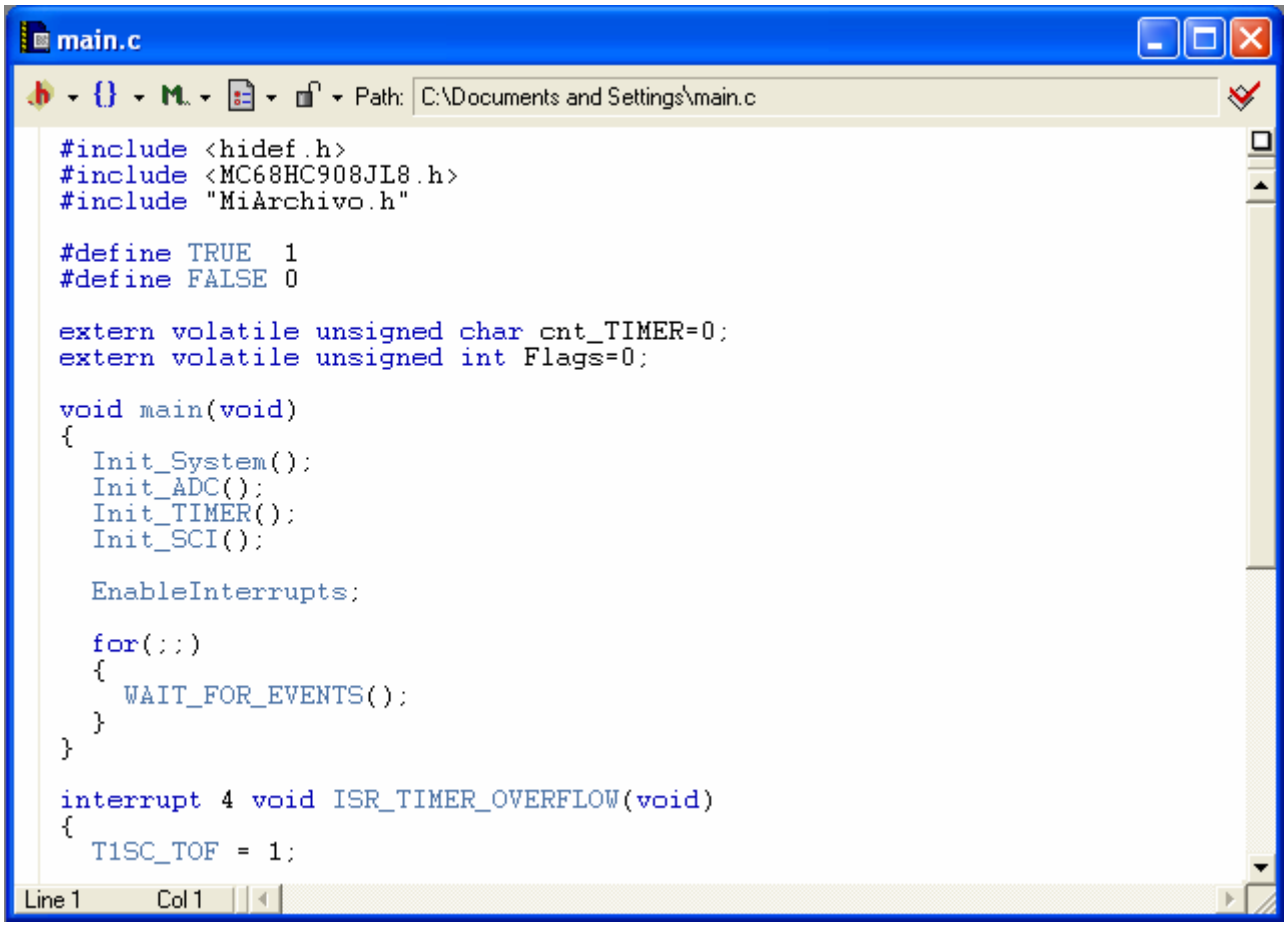
```
int suma;
suma = A + B;
if (suma >= 10)
    printf (“Mayor que…”);
else
    printf (“Menor que…”);
```

```
move.w (A0)+,D0
add.w    (A0),D0
cmp.w   #10,D0
bgt       Print_Msg_A
bra       Print_Msg_B
…
Print_Msg_A:
…
```

```
01000100010100…
10101001001010…
01001001000101…
11011101010101…
…
```

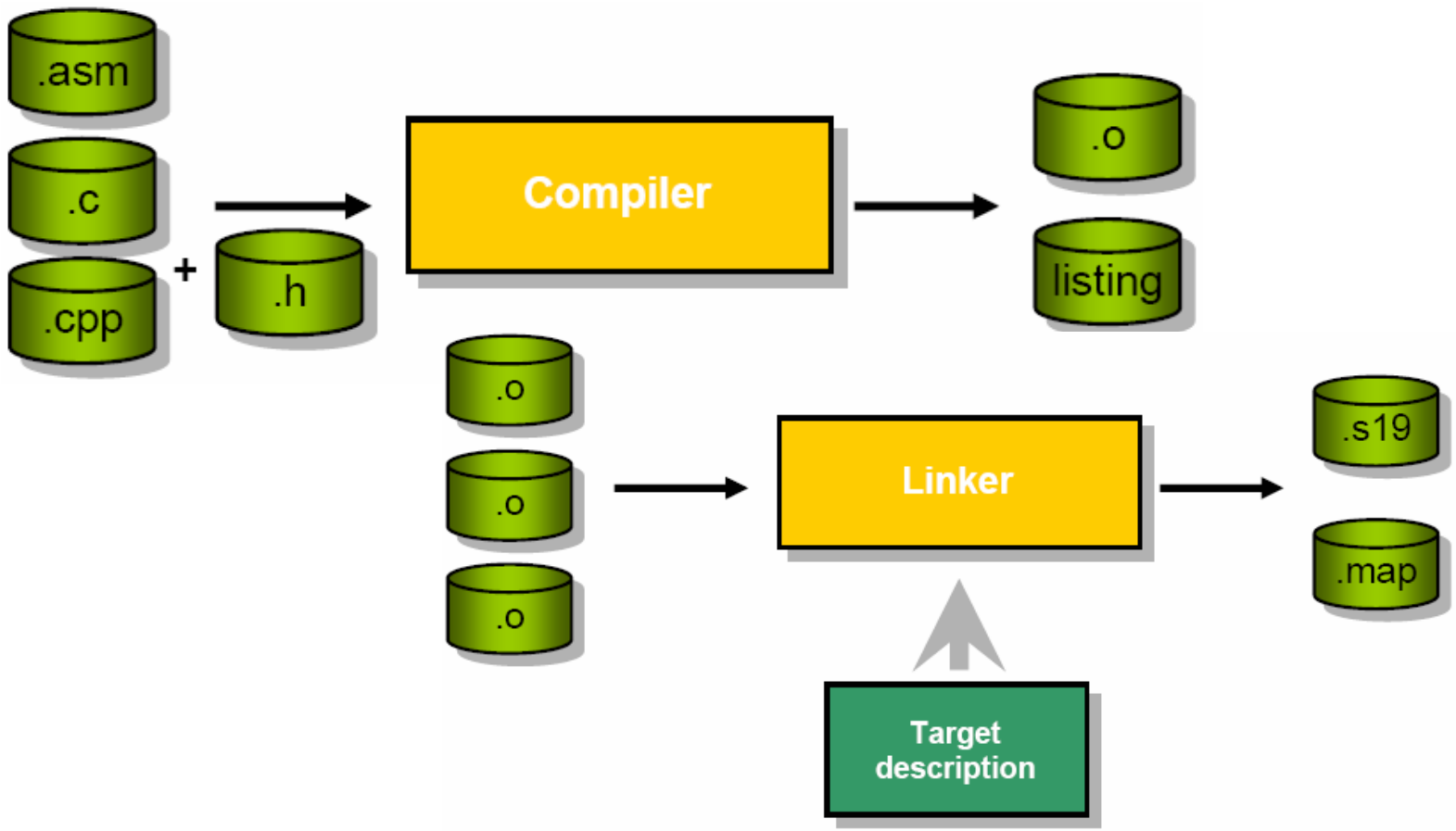


Desarrollo de Firmware



Desarrollo de Firmware

- Compiladores y Enlazadores.

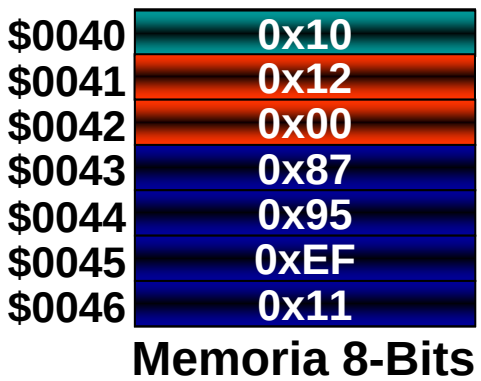


Desarrollo de Firmware

- Lenguaje C para Sistemas Embebidos.
  - El Preprocesador.
  - Variables y Tipos de Datos.

```
unsigned char X = 0x10;
unsigned int  A = 0x1200;
unsigned long B = 0x8795EF11;
```

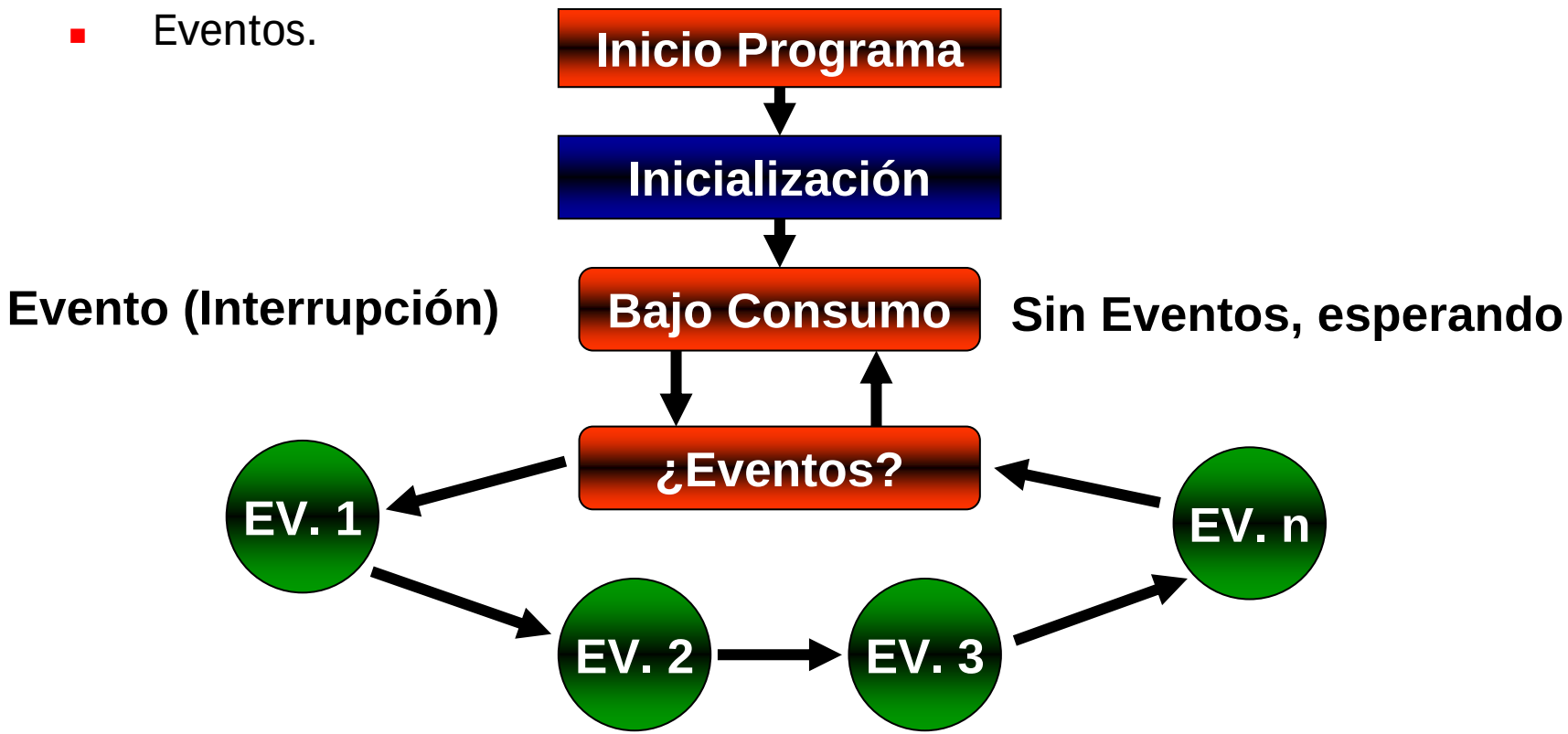
Tipo de Dato **Variable y Valor Ini.**

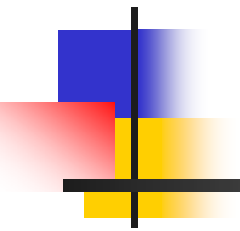


  - Modificadores: const, extern, volatile, static.
  - Arreglos, estructuras de datos, punteros.

Desarrollo de Firmware

- Lenguaje C para Sistemas Embebidos.
  - Funciones en C (Subrutinas y Funciones).
  - Eventos.





Unidades de procesamiento

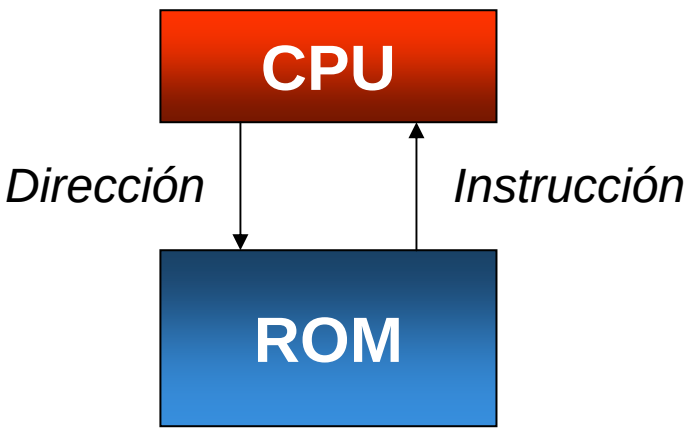
Microprocesadores, instrucciones, modos de direccionamiento

Características de la CPU para S.E:

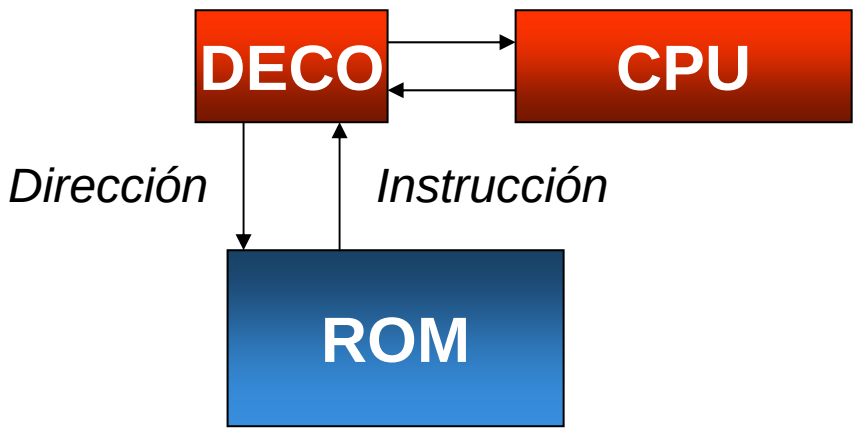
- Eficiencia, tamaño de código (sistemas no complejos):
  - Sistemas típicamente sin disco duro y/o diseños con poca cantidad de memoria RAM y ROM.
  - **CISC**
    - Buena opción, eficiencia en el tamaño del código.
    - Una instrucción realiza varias operaciones.
    - Memoria embebida dentro del mismo chip, más costosa.
  - ***RISC***: Optimizada para velocidad.
    - **CISC** para la programación, **RISC** interno.

Características de la CPU para S.E:

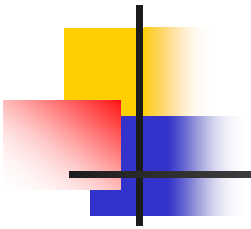
- Eficiencia, tamaño de código:
  - Técnicas de compresión de instrucciones.
  - Eficiencia en tiempo de Ejecución:



Esquema tradicional



Compresión de instrucciones

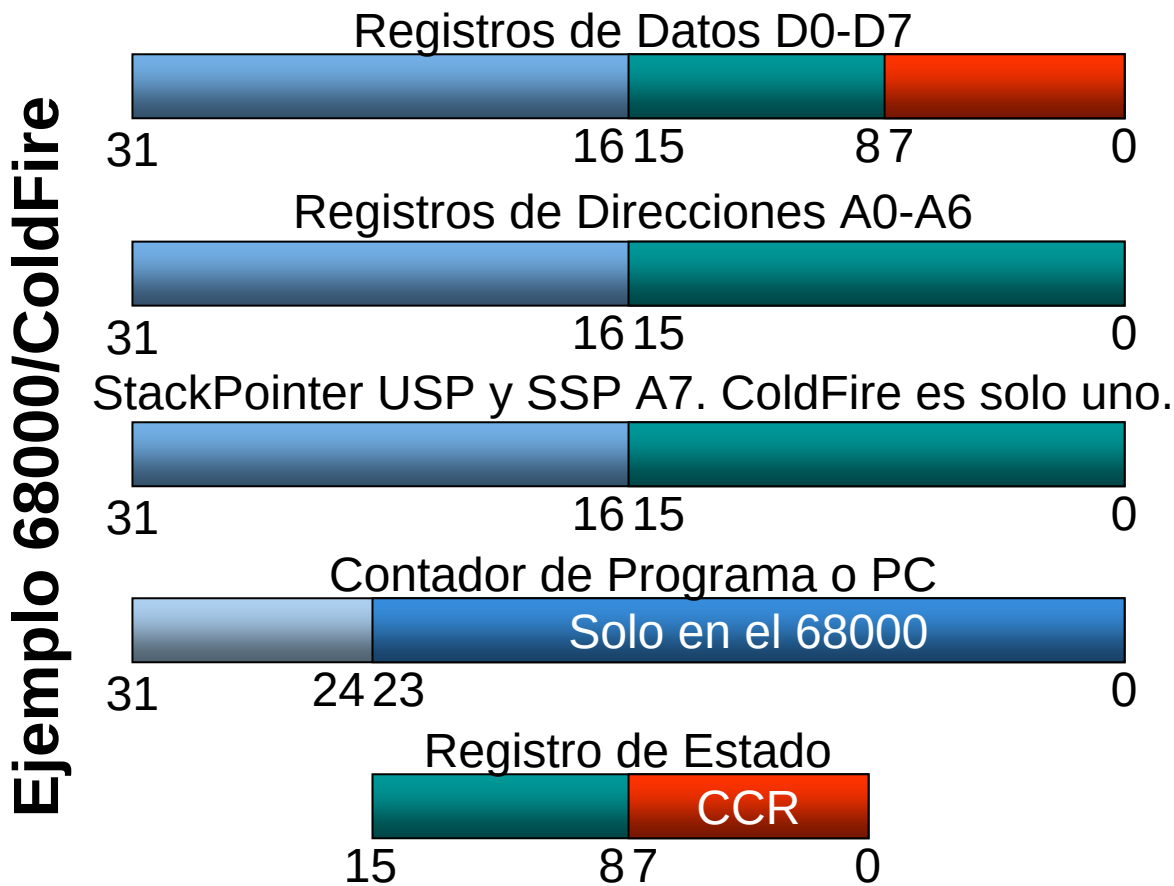


Microprocesadores

- Difieren de los procesadores tradicionales como: INTEL x86, AMD, AMD 64, entre otros.
- Son orientados a aplicaciones específicas.
- Eficiencia:
  - **Bajo consumo de potencia**: DPM y DVS.
  - **Código pequeño**: ISA optimizada.
  - **Módulos**: Apropiados para S.E.
    - **Conversor** A/D, D/A, TIMER, PWM, Serial.
    - **Avanzados**: DRAM, DMA, LCD Gráfico, Interrupciones.

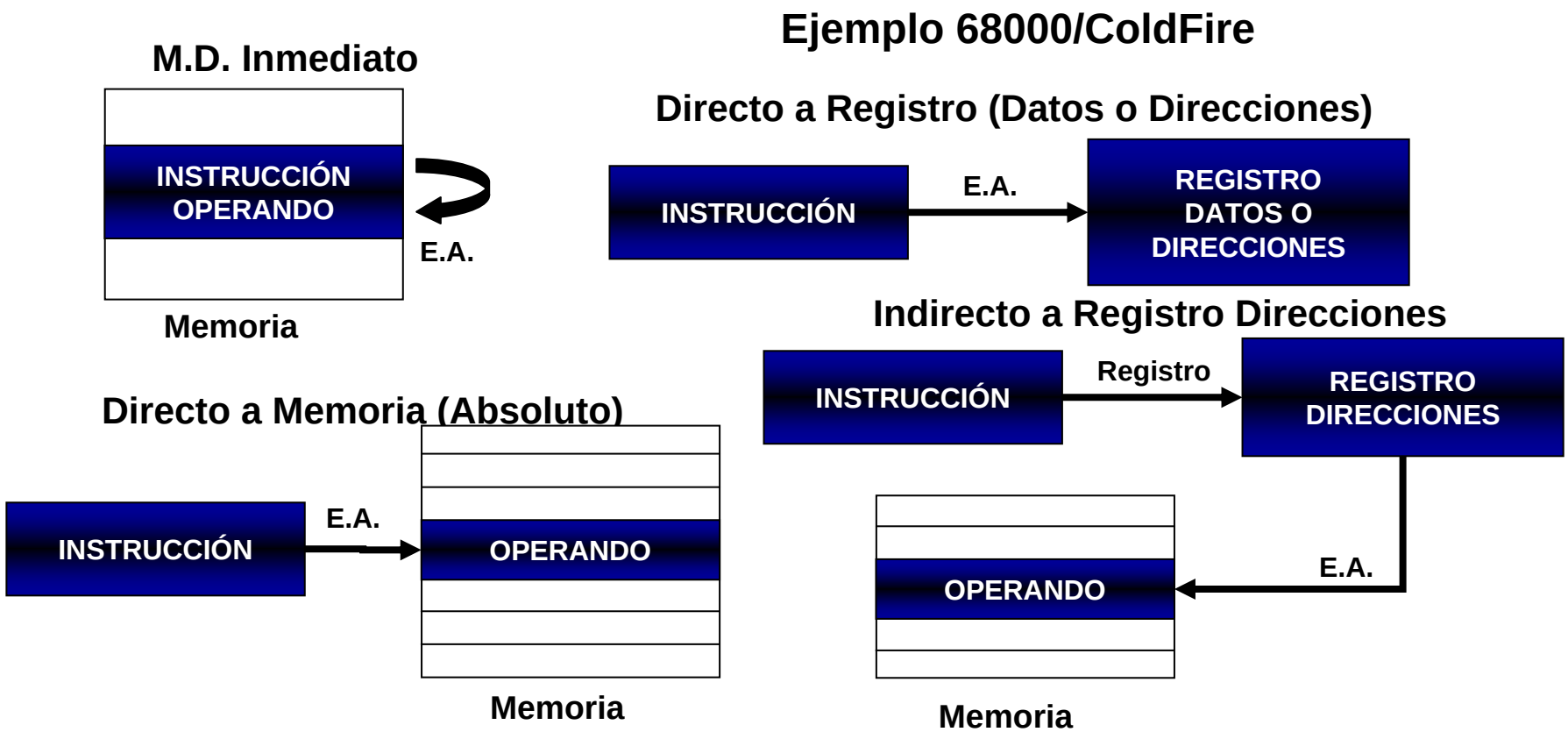
Modelo de Programación

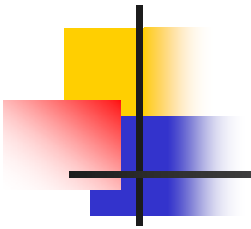
Conjunto de elementos ofrecidos al usuario para la programación.



Modos de Direccionamiento

Conjunto de mecanismos para proveer de datos a las instrucciones

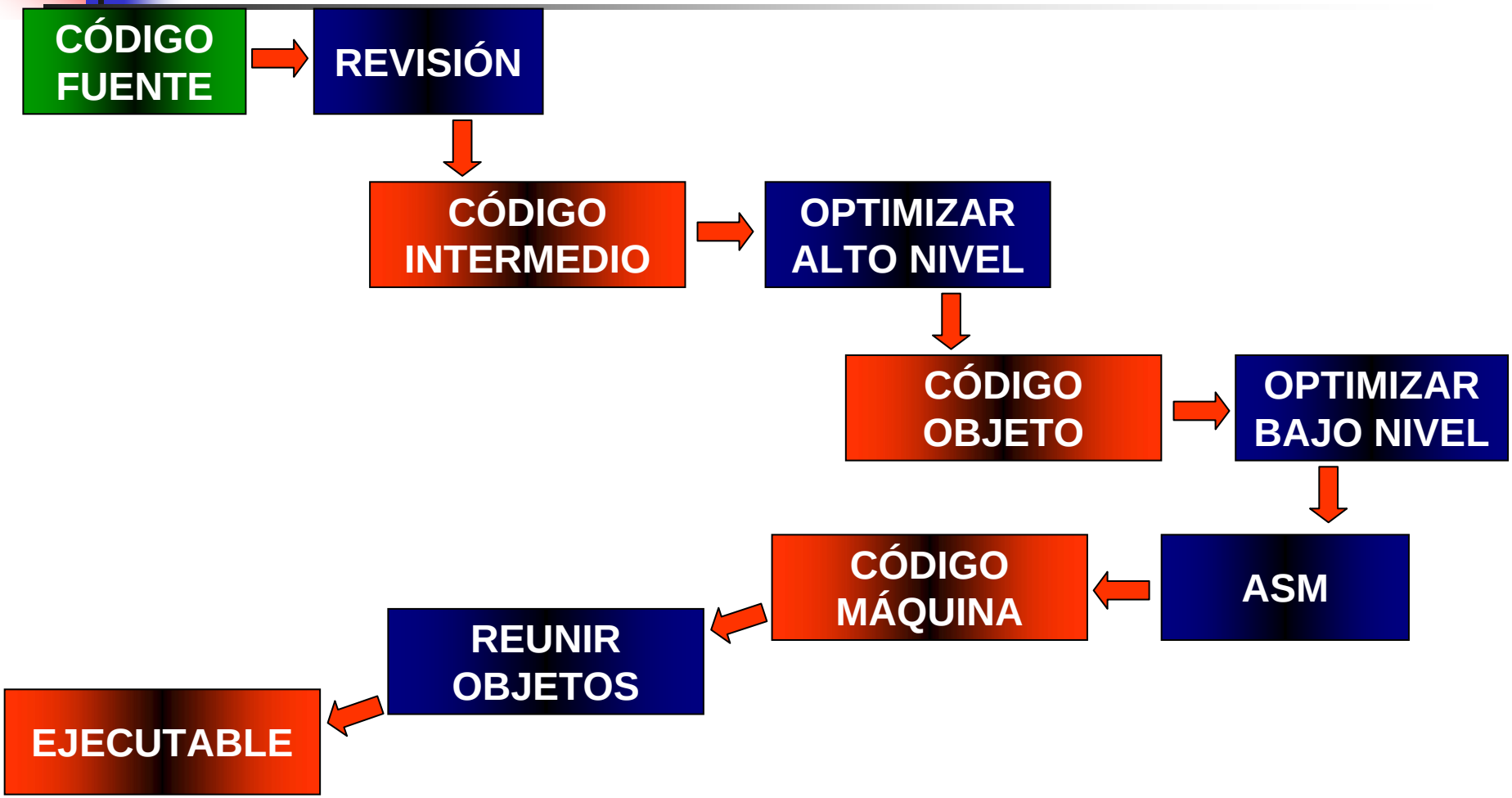




Lenguaje C para S.E.

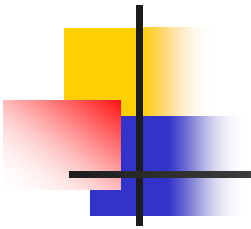
- Código Fuente.
  - Se escribe en un lenguaje de alto nivel: C.
  - Más cercano a los humanos, menos a la máquina.
    - Debe ser traducido a lenguaje de máquina.
    - Los pasos son:
      - **Revisión**: Sintaxis y semántica.
      - **Traducción** lenguaje intermedio.
      - **Optimizaciones** de alto nivel.
      - Llevar a **código objeto**: dependiente de la máquina.
      - **Optimizaciones** de bajo nivel.
      - **Reunión** con otros módulos: Código ejecutable.

Pasos del código fuente al binario



El compilador

- Toma el código fuente (.c y .h) y lo organiza de acuerdo a las directivas del preprocesador.
- Revisión de sintaxis y semántica.
- Traducción a código objeto con las respectivas optimizaciones.

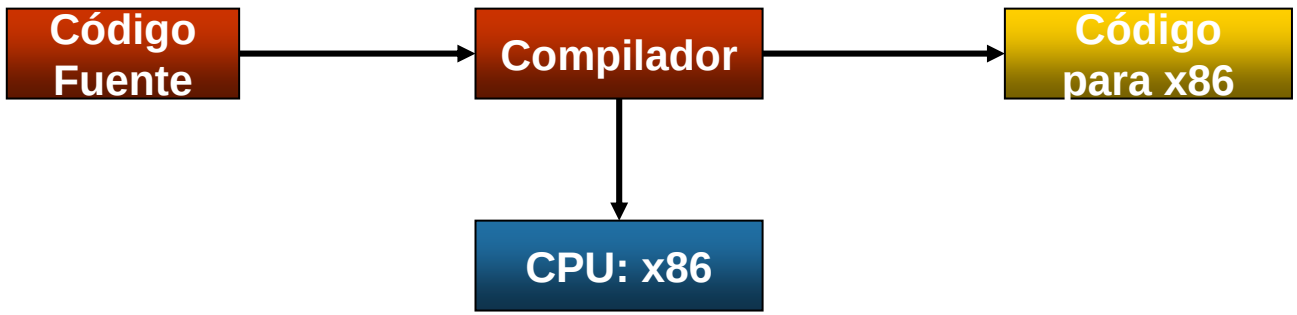


Compiladores

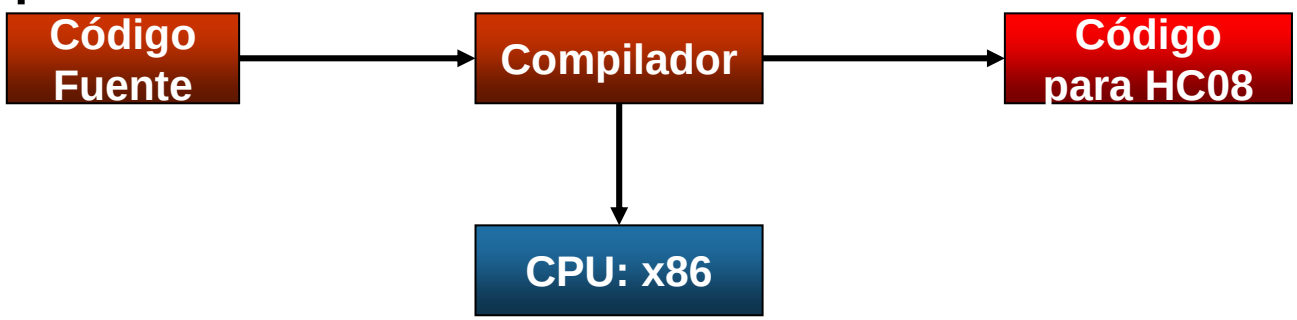
- Compiladores cruzados
  - Compilador tradicional
    - Se ejecuta en una arquitectura específica (x86).
    - Genera código para esa misma arquitectura.
    - Visual C++ (x86). Genera código para x86.
  - Compilador cruzado
    - Se ejecuta en una arquitectura específica (x86).
    - Genera código para otra arquitectura (m68k, arm).
    - gcc-m68k-elf: Compilador de C que corre en x86 pero genera código para un Motorola 68000.

Esquema de los compiladores

- Compiladores cruzados
  - Compilador tradicional

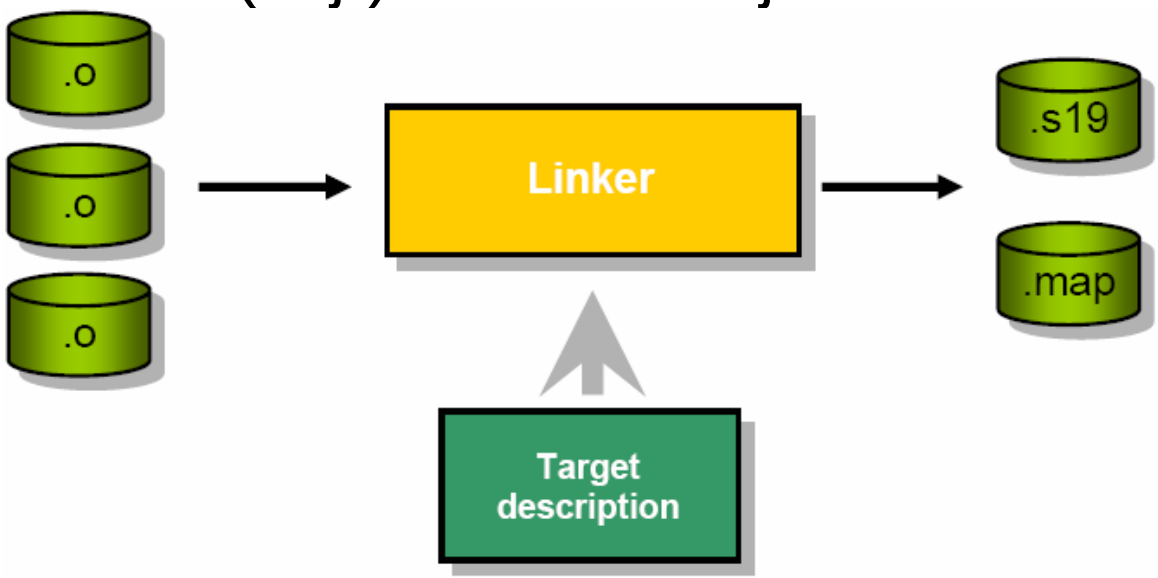


  - Compilador cruzado



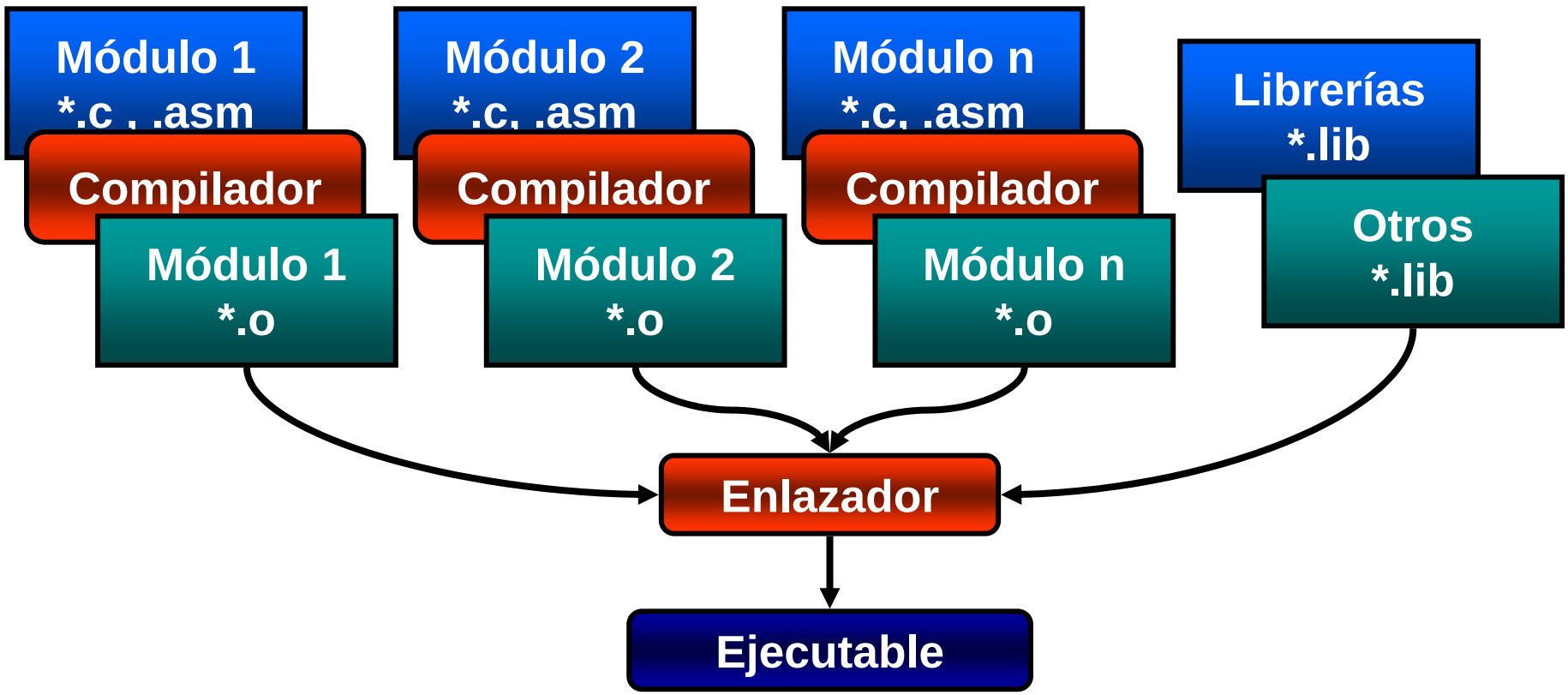
Ligadores

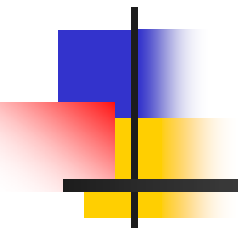
- Enlazador: Linker
  - Varios módulos: varios archivos objeto.
  - Resuelve las referencias externas.
  - Reúne todos (obj.) en un solo ejecutable.



Binario

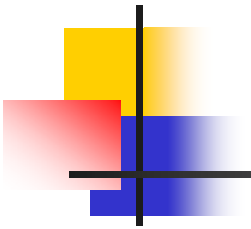
- Binario o ejecutable:





Arquitectura ARM

Advanced Risc Machine



Acerca de la arquitectura ARM

The ARM architecture has been designed to allow very small, yet high-performance implementations. The architectural simplicity of ARM processors leads to very small implementations, and small implementations allow devices with very low power consumption.

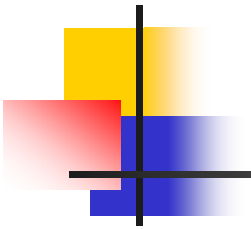
The ARM is a *Reduced Instruction Set Computer* (RISC), as it incorporates these typical RISC architecture features:

- a large uniform register file
- a *load/store* architecture, where data-processing operations only operate on register contents, not directly on memory contents
- simple addressing modes, with all load/store addresses being determined from register contents and instruction fields only
- uniform and fixed-length instruction fields, to simplify instruction decode.

In addition, the ARM architecture gives you:

- control over both the *Arithmetic Logic Unit* (ALU) and shifter in every data-processing instruction to maximize the use of an ALU and a shifter
- auto-increment and auto-decrement addressing modes to optimize program loops
- Load and Store Multiple instructions to maximize data throughput
- conditional execution of all instructions to maximize execution throughput.

These enhancements to a basic RISC architecture allow ARM processors to achieve a good balance of high performance, low code size, low power consumption and low silicon area.



Registros del ARM

ARM has 31 general-purpose 32-bit registers. At any one time, 16 of these registers are visible. The other registers are used to speed up exception processing. All the register specifiers in ARM instructions can address any of the 16 visible registers.

The main bank of 16 registers is used by all unprivileged code. These are the User mode registers. User mode is different from all other modes as it is unprivileged, which means:

- User mode is the only mode which cannot switch to another processor mode without generating an exception
- memory systems and coprocessors might allow User mode less access to memory and coprocessor functionality than a privileged mode.

Registros del ARM

Two of the 16 visible registers have special roles:

Link register — Register 14 is the *Link Register* (LR). This register holds the address of the next instruction after a Branch and Link (BL) instruction, which is the instruction used to make a subroutine call. At all other times, R14 can be used as a general-purpose register.

Program counter — Register 15 is the *Program Counter* (PC). It can be used in most instructions as a pointer to the instruction which is two instructions after the instruction being executed. All ARM instructions are four bytes long (one 32-bit word) and are always aligned on a word boundary. This means that the bottom two bits of the PC are always zero, and therefore the PC contains only 30 non-constant bits.

The remaining 14 registers have no special hardware purpose. Their uses are defined purely by software. Software normally uses R13 as a *Stack Pointer* (SP).

Organización de los registros

Modes						
	Privileged modes					
		Exception modes				
User	System	Supervisor	Abort	Undefined	Interrupt	Fast interrupt
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

Referencias

- Luís Germán García Morales, “Sistemas Embebidos – notas de clase-”, Universidad de Antioquia, Grupo de Microelectrónica y Control.
- ARM Manual de referencia de la arquitectura