uc3m

Departamento Tecnología Electrónica

dte UC3M

Tema 2: Arquitectura Interna de una CPU

Sistemas Digitales Basados en Microprocesadores

Universidad Carlos III de Madrid

Dpto. Tecnología Electrónica

Nota 1: Las figuras utilizadas para ilustrar las características de los productos de ARM se han obtenido de las publicaciones técnicas disponibles en: https://developer.arm.com/ip-products/processors

Nota 2: Las figuras utilizadas para ilustrar los productos de STMicroelectronics se han obtenido de las publicaciones técnicas disponibles en: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html

Índice

Tipos de Arquitecturas

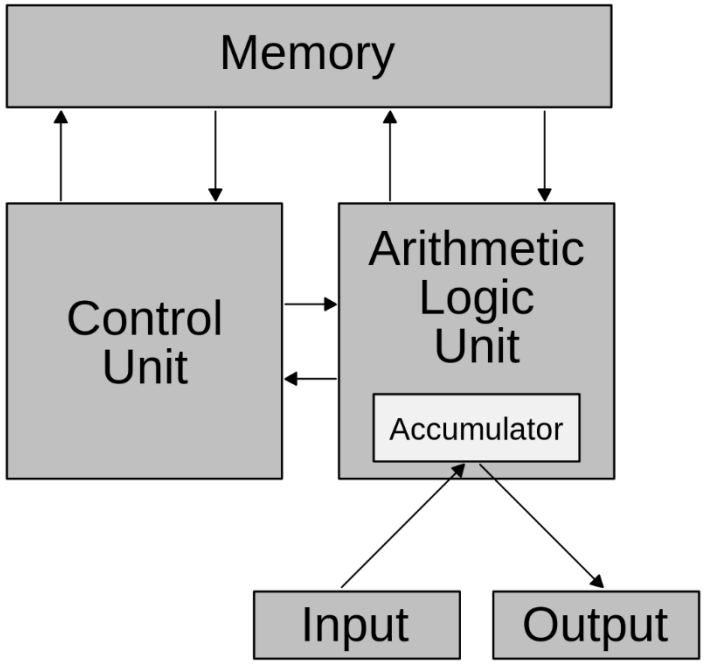
Arquitectura Von Neumann

- John Von Neumann, en su artículo del año 1945, definió una computadora de propósito general basada en la idea de **programa almacenado**
- Los componentes principales eran:
  - Una memoria principal
    - Almacenaba tanto datos como instrucciones
  - Una unidad de cálculo para operaciones aritméticas y lógicas
    - Lo que se conoce como una ALU
  - Una unidad de control
    - Que interpreta las instrucciones obtenidas de la memoria y las ejecuta
  - Un equipamiento de entrada/salida
    - Para interactuar con el mundo exterior
- Esto lo plasmó en una máquina denominada IAS (*Institute for Advanced Study machine*)

Arquitectura Von Neumann

- Memoria común para datos e instrucciones
  - 1000 palabras de 40 bits
    - Datos: Números binarios con signo
    - Instrucciones:
      - Cada palabra tenía 2 instrucciones de 20 bits
      - Cada instrucción tiene
        - Código de operación de 8 bits
        - Dirección codificada en 12 bits

https://live.staticflickr.com/3749/11239892036_e45a931251_b.jpg



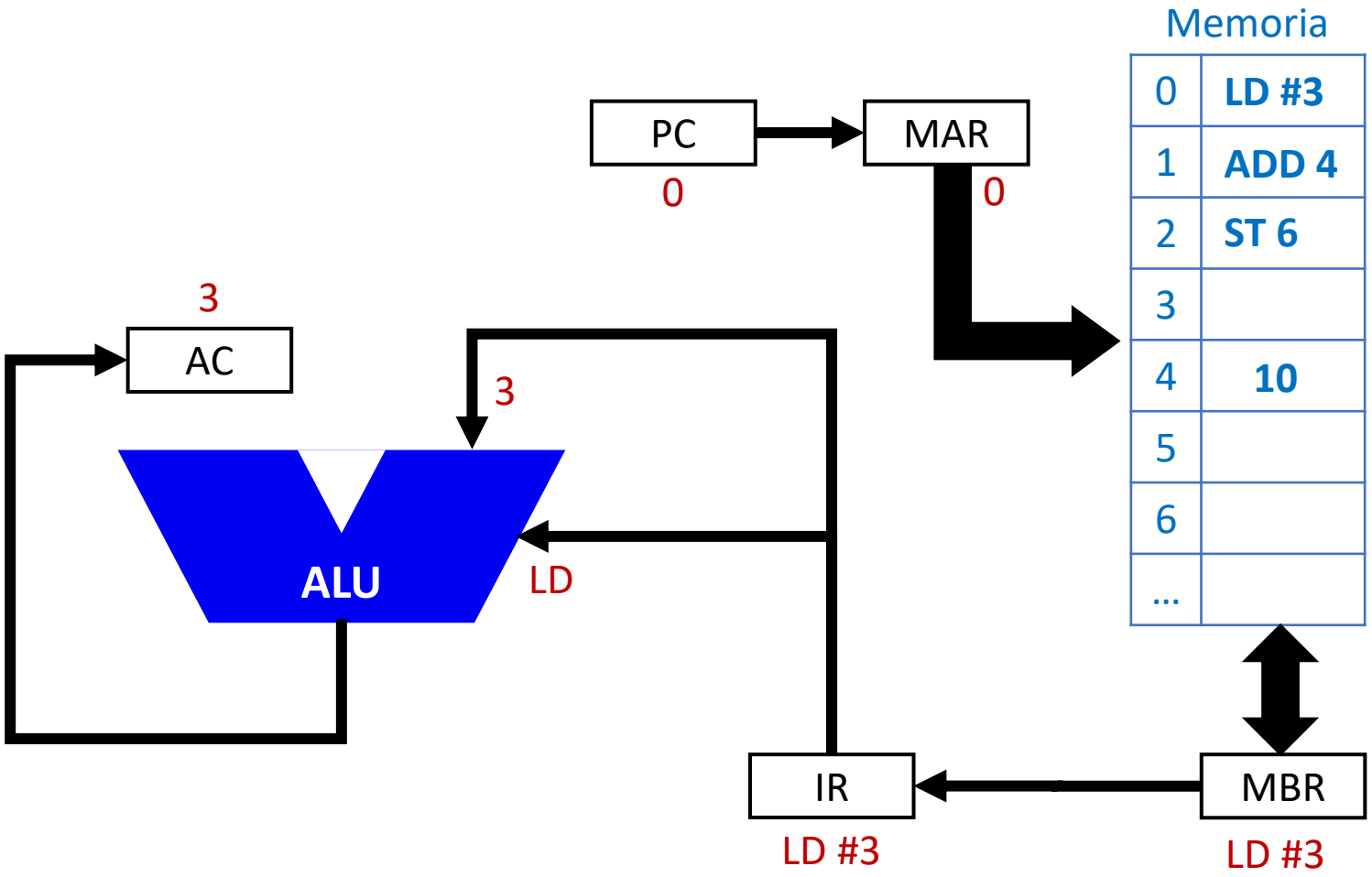
https://upload.wikimedia.org/wikipedia/commons/thumb/8/84/Von_Neumann_architecture.svg/1075px-Von_Neumann_architecture.svg.png

Arquitectura Von Neumann

- El IAS contaba con 21 instrucciones que se podían agrupar en los siguientes tipos:
  - Transferencia de Datos
  - Desvíos Incondicionales
  - Desvíos Condicionales
  - Aritméticas y Lógicas
- También describió el modo de funcionamiento de la Unidad de Control
  - 1.- La UC captura la instrucción de la memoria
  - 2.- La decodifica
  - 3.- La ejecuta y vuelve al paso 1 para capturar la siguiente instrucción en memoria
    - Es decir, la máquina de Von Neumann seguía una ejecución secuencial de las instrucciones, que se colocaban de forma lineal en la memoria, alterándose dicha linealidad sólo por la existencia de instrucciones de desvíos (condicionales e incondicionales)

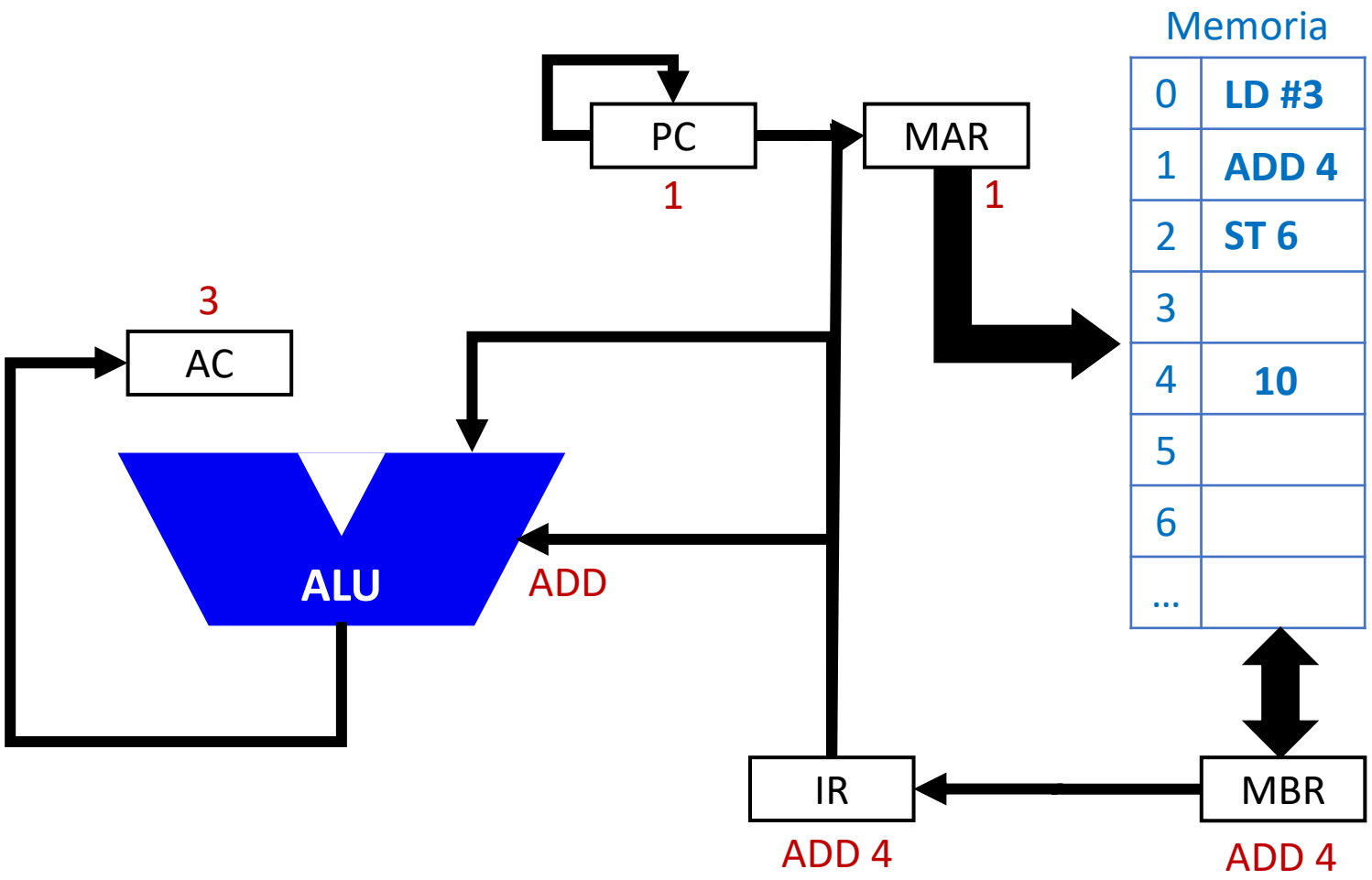
Arquitectura Von Neumann

- Ilustremos los conceptos de Von Neumann con un ejemplo sencillo
  - 1º Ciclo



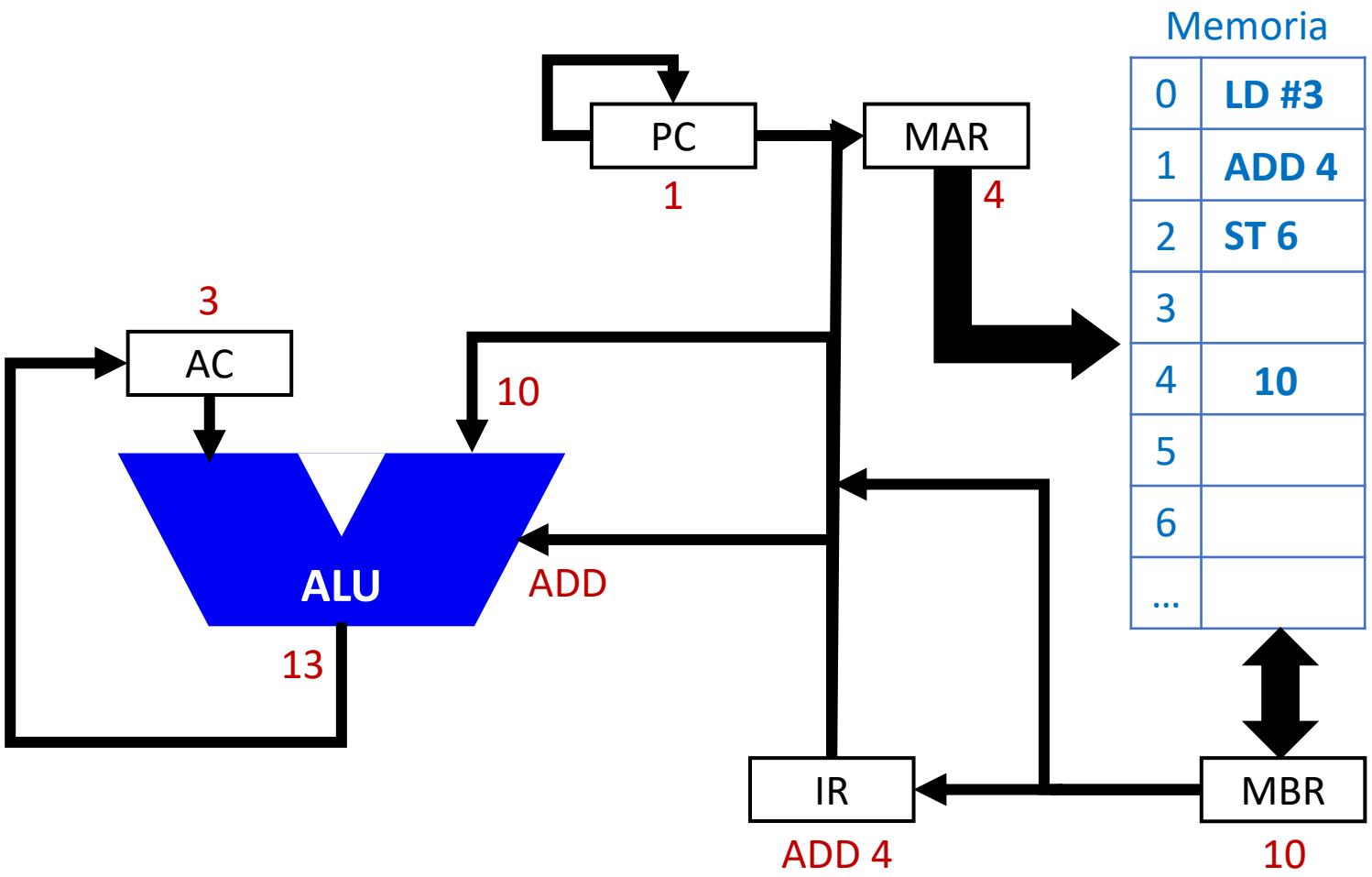
Arquitectura Von Neumann

- 2º Ciclo



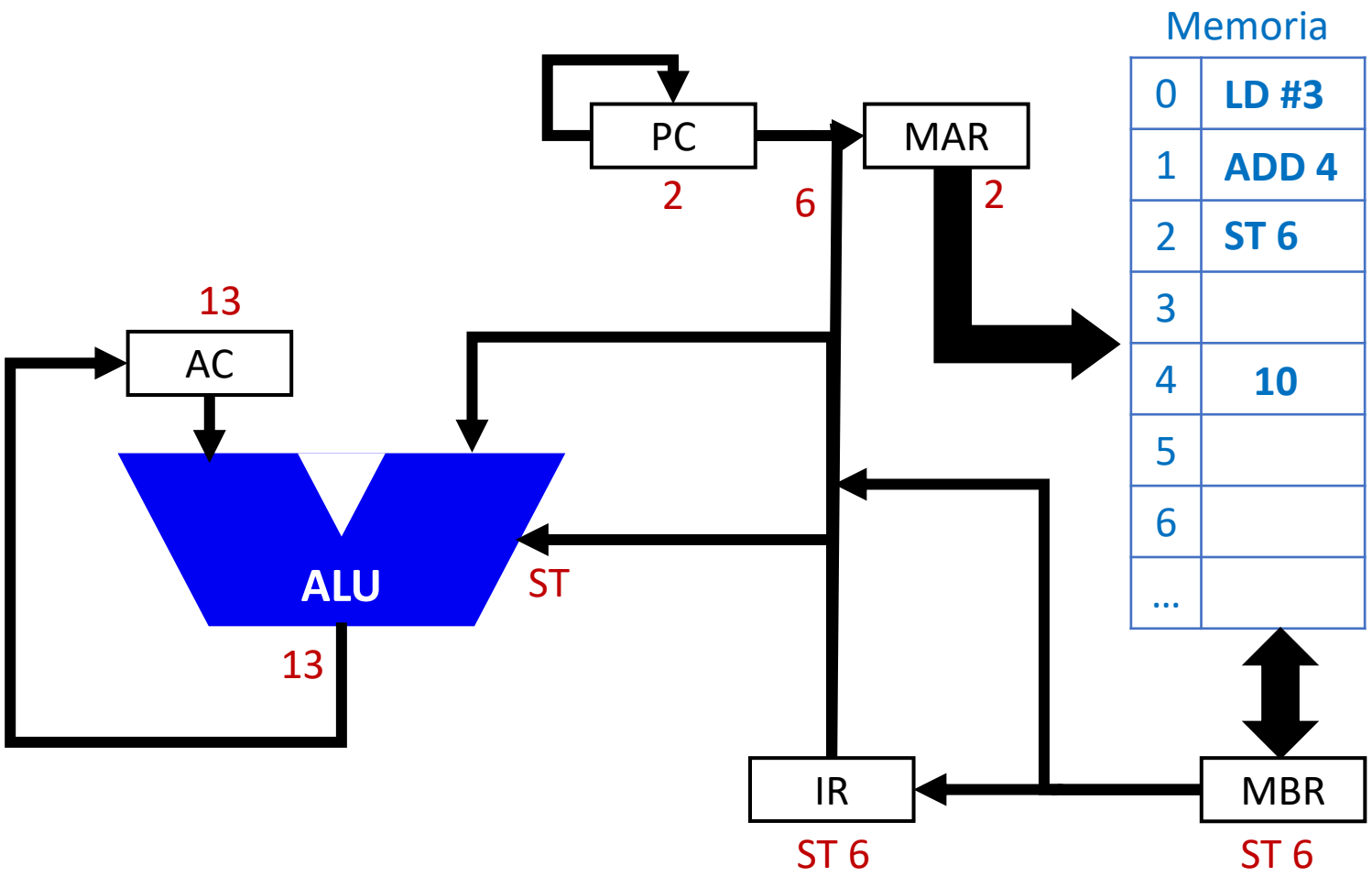
Arquitectura Von Neumann

- 3º Ciclo



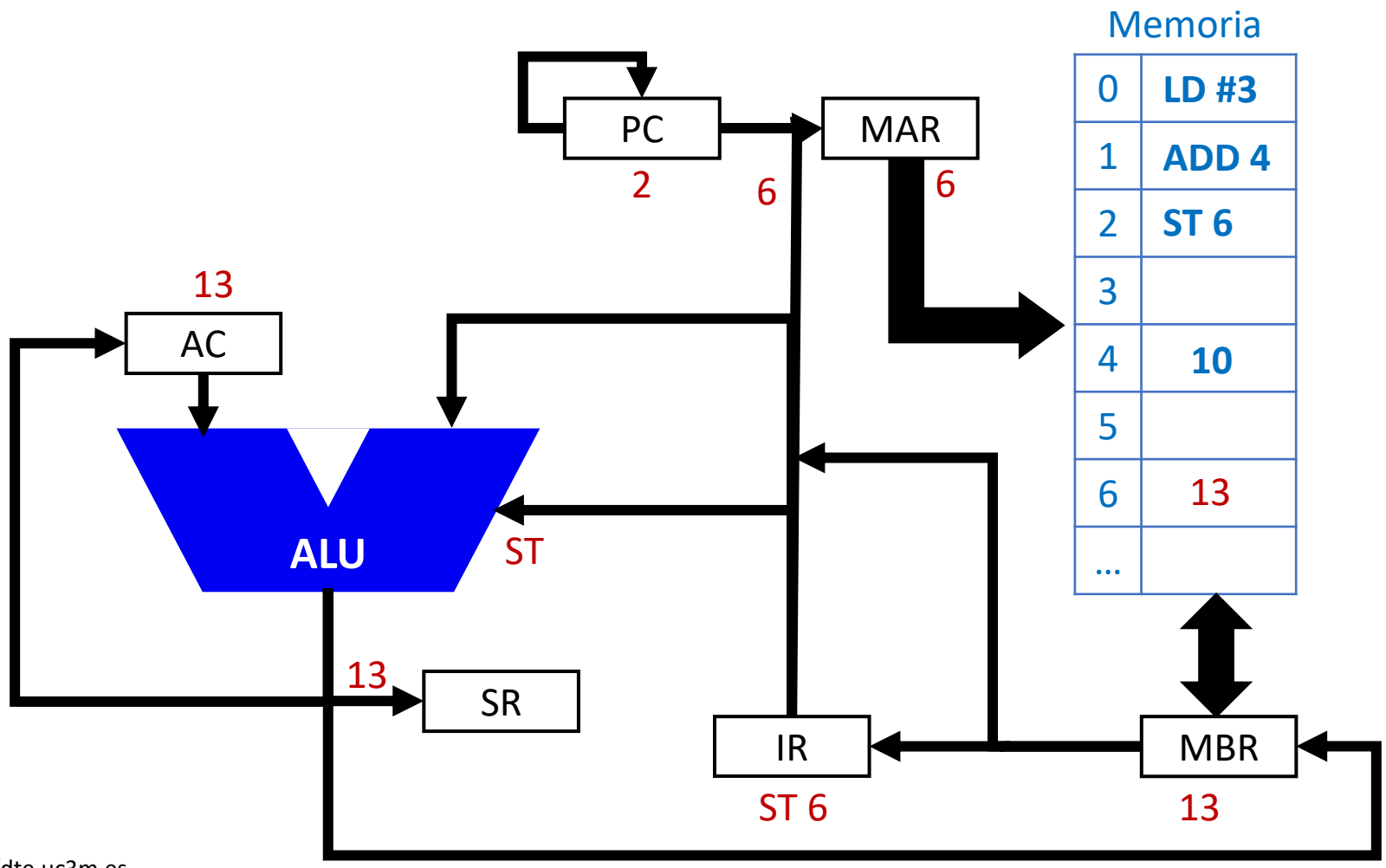
Arquitectura Von Neumann

- 4º Ciclo



Arquitectura Von Neumann

- 4º Ciclo



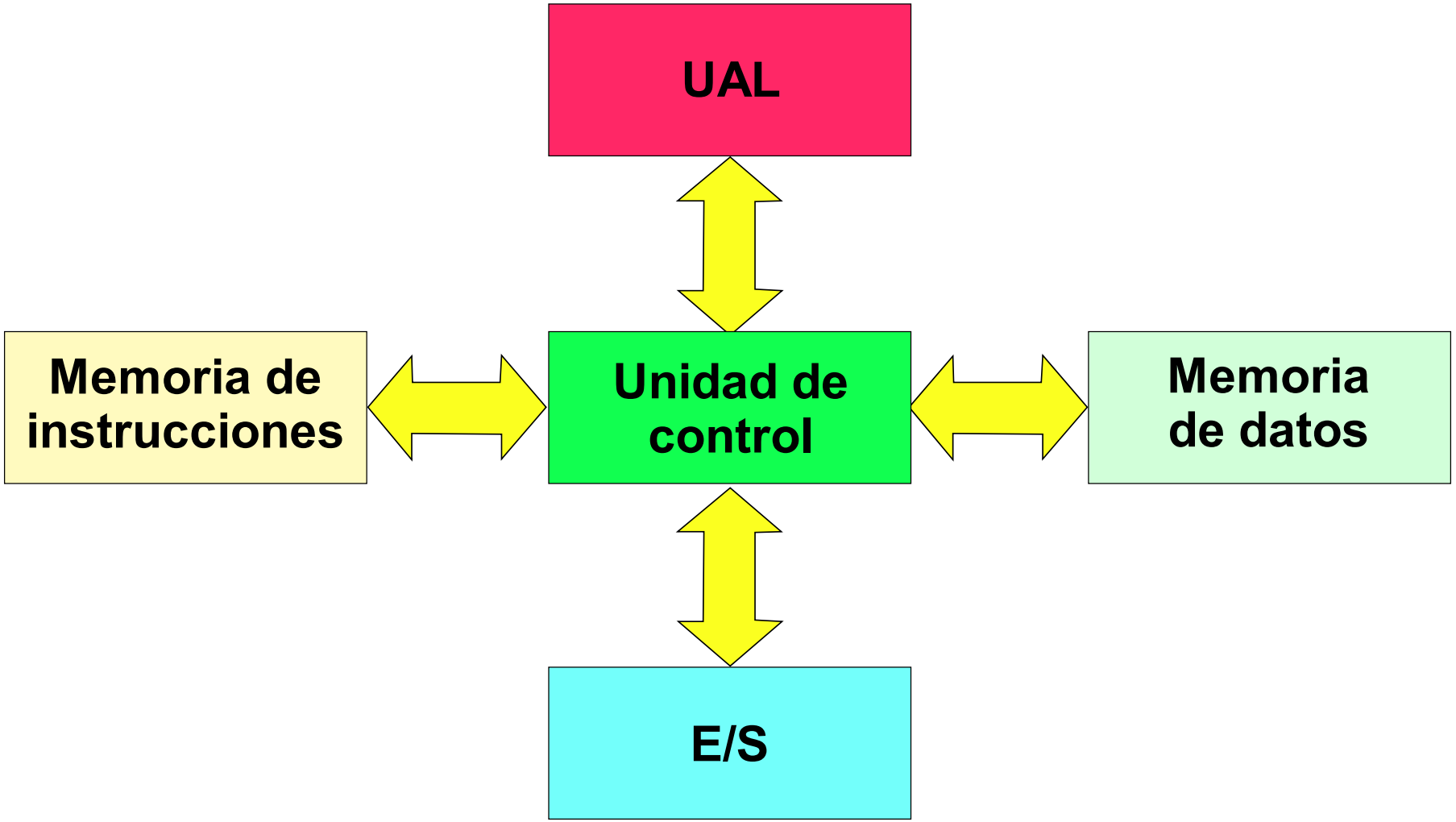
Arquitectura Von Neumann

- Ampliaciones posteriores han dado lugar a dos tipos de arquitecturas:
  - Basada en Acumulador:
    - Es la original de Von Neumann (aunque a día de hoy pueden tener más de un acumulador)
    - Casi toda operación tiene como fuente o como destino el acumulador
  - Basada en Registros:
    - Surge para mejorar prestaciones:
      - Las operaciones entre registros son más rápidas que cuando hay que consultar a memoria
      - Cuantos más registros se tengan, menos accesos a memoria son necesarios en operaciones iterativas
    - Se sustituye el acumulador por un conjunto de registros (su número depende de la CPU concreta)
    - Los registros pueden tener uso indistinto o específico:
      - De Propósito General
      - Sólo de datos
      - De direcciones
    - En algunas arquitecturas se fuerza a que todas las operaciones se hagan sólo entre registros (salvo las de transferencia de datos)

Arquitectura Harvard

- Se elimina el concepto de Memoria Principal. En esta arquitectura existe:
  - Una memoria exclusivamente para datos
  - Una memoria exclusivamente para instrucciones
  - Buses (tanto de datos, como de direcciones) diferenciados para cada una de las memorias
    - Sus números de líneas pueden ser distintos
      - El tamaño de palabra de datos y de instrucciones puede ser distinto
      - La capacidad de las memorias pueden ser distintas
- Ventajas:
  - Se incrementa la capacidad de direccionamiento
  - Se pueden adaptar mejor a las necesidades de las aplicaciones objetivo de dicha CPU
  - Se incrementa la fiabilidad de las aplicaciones, por garantía de integridad del código
- Inconvenientes:
  - Interfaz Externa más compleja y conexionado más amplio

Arquitectura Harvard



https://upload.wikimedia.org/wikipedia/commons/3/38/Harvard_architecture-es.svg

Arranque de la CPU

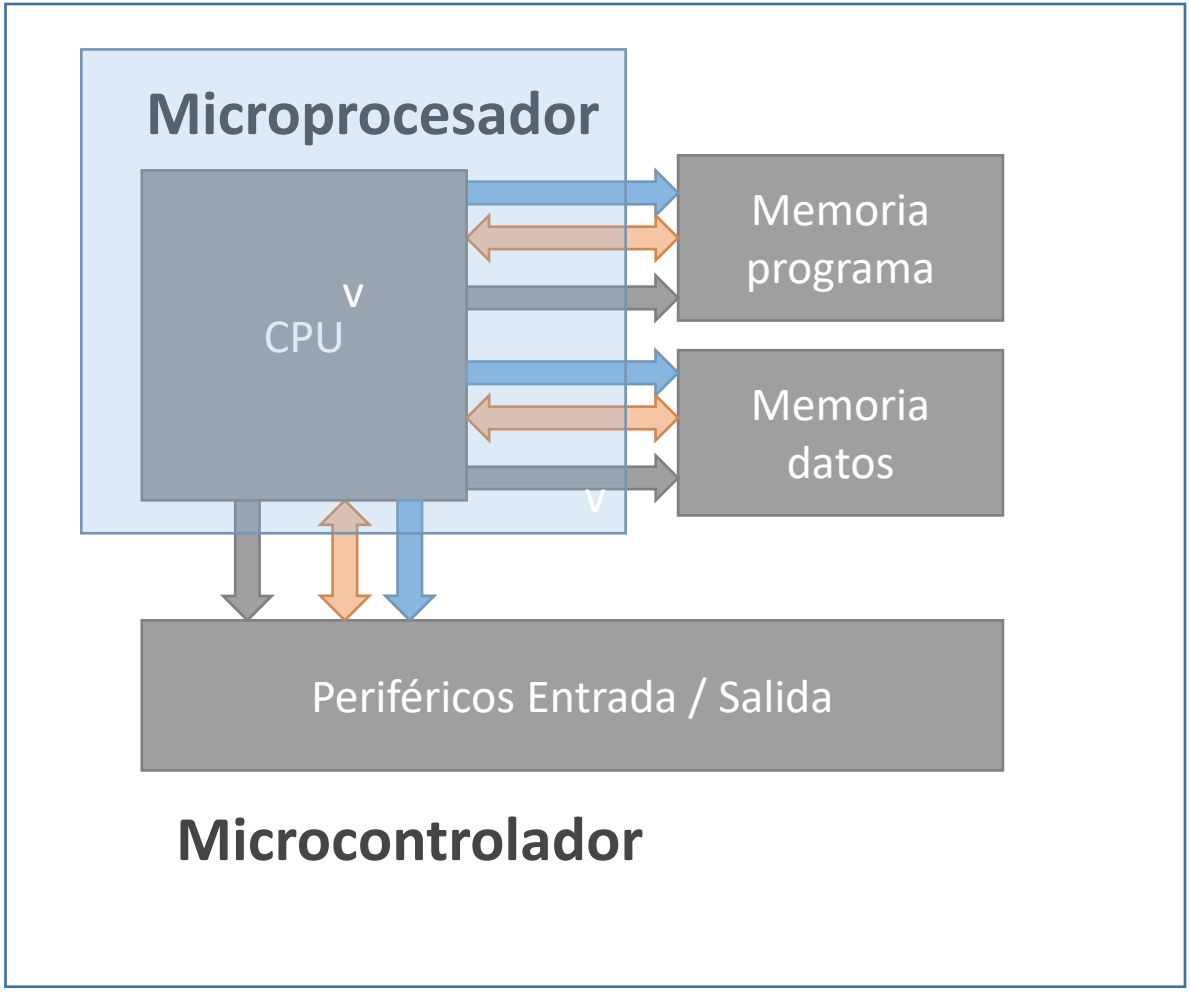
- La CPU tiene que ser un elemento determinista, por lo que hay que garantizar que sus condiciones iniciales son siempre las mismas:
  - Tiene que tener un valor estable en el SR
  - Tiene que saber el valor inicial del PC
  - Tiene que saber el valor inicial del SP, si existe
  - Y en muchos casos el valor inicial de otros muchos registros
- Dependiendo de la CPU esos valores son fijos, o pueden ser modificados por el programador
  - En ese caso se graba el valor en una posición de memoria que al inicio es accedida por la CPU para obtener su valor (vector de reset, etc.)

Microprocesador vs. Microcontrolador

- Un Microprocesador es un circuito integrado que contenga todos los elementos de control de una máquina de calcular:
  - Unidad Aritmética Lógica (ALU)
  - Unidad de Control
  - Registros internos para el flujo por la ruta de datos:
    - PC, IR, MAR, MBR, SR, SP, etc.

- Un Microcontrolador es un chip que, además de tener un Microprocesador, contiene:
  - Memoria(s)
  - Dispositivos de E/S

Microprocesador vs. Microcontrolador



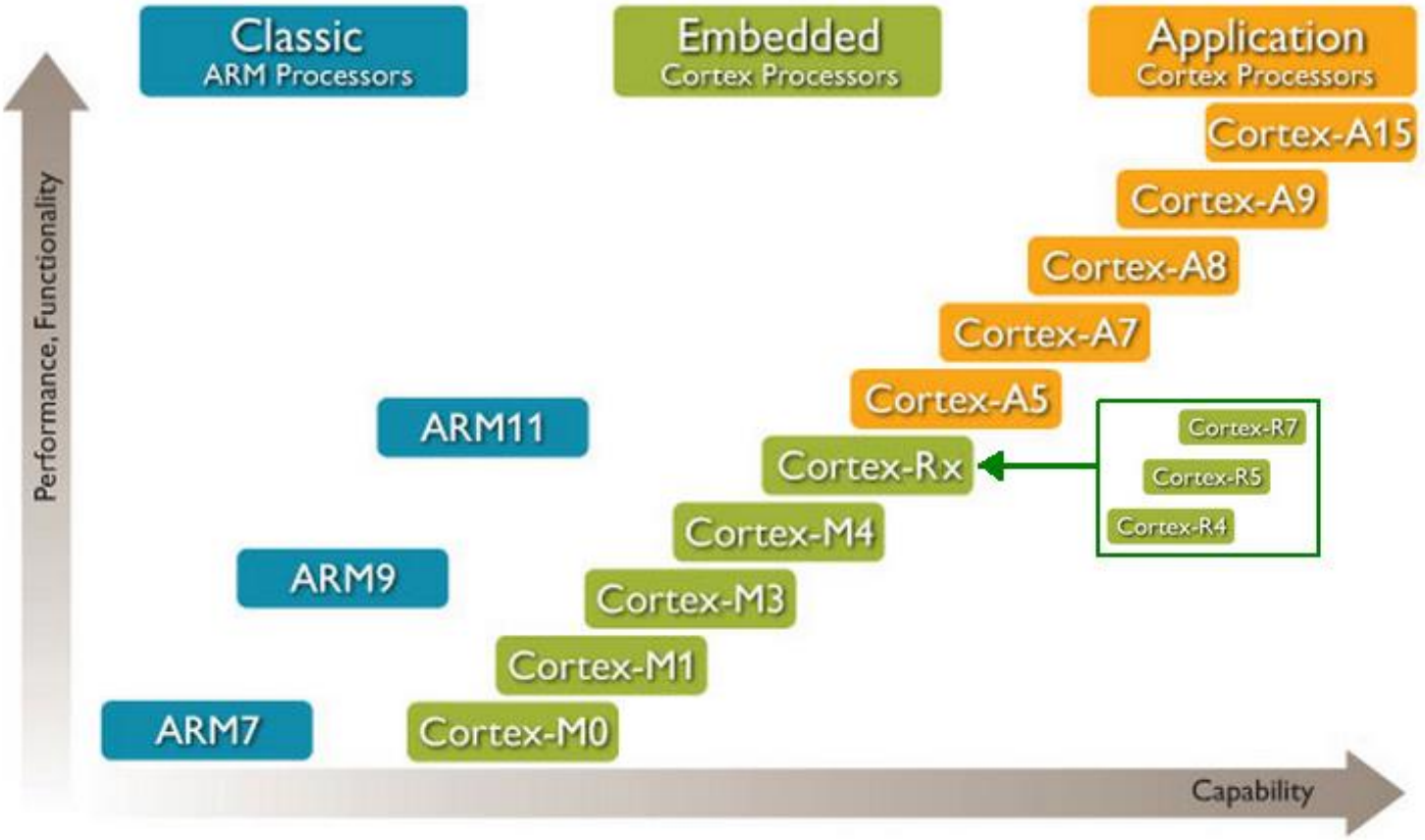
La Familia ARM

La Familia ARM

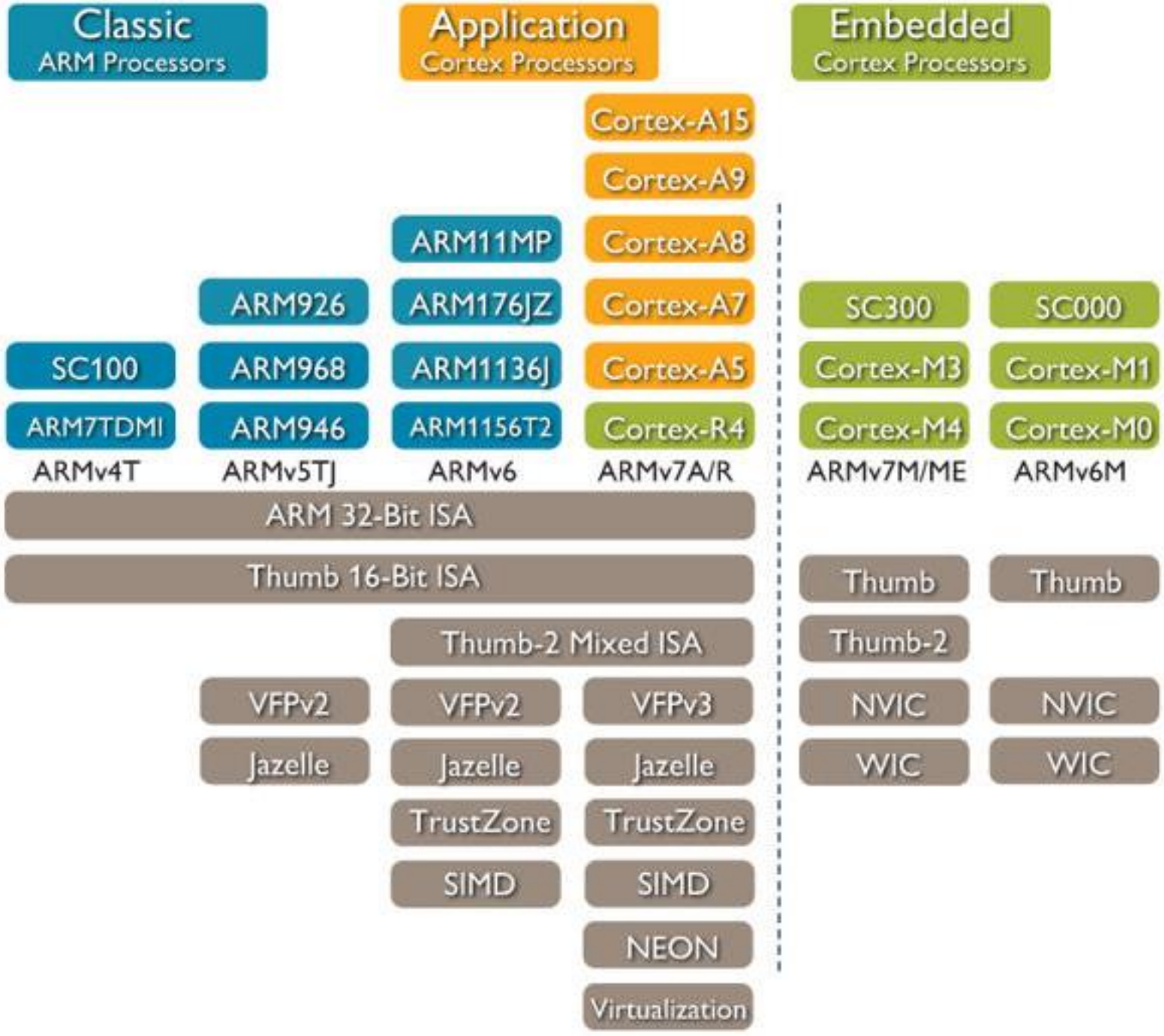
- ARM Holdings plc es una multinacional dedicada a los semiconductores y al desarrollo de software con sede en Cambridge, Reino Unido.
- Es considerada la empresa dominante en el campo de los chips de smartphones y tablets.
- La compañía fue fundada como Advanced RISC Machines (ARM) en 1990 por Robin Saxby
  - Una empresa conjunta entre Acorn Computers (desde 1983), Apple Computer (ahora Apple Inc.) y VLSI Technology.
- Pretende promover el desarrollo de la arquitectura ARM, la cual fue utilizada en su origen en el Acorn Archimedes y fue seleccionado por Apple para su proyecto Apple Newton.
- Cronología:
  - 1981 – ARM1
  - 1986 – ARM2: primera versión comercial (32 bits), con mejor rendimiento que el 286
  - 1991 – ARM6: Apple Newton y RiscPC
  - 1994 – ARM7TDMI: millones de unidades en teléfonos móviles y videojuegos (2009)

La Familia ARM

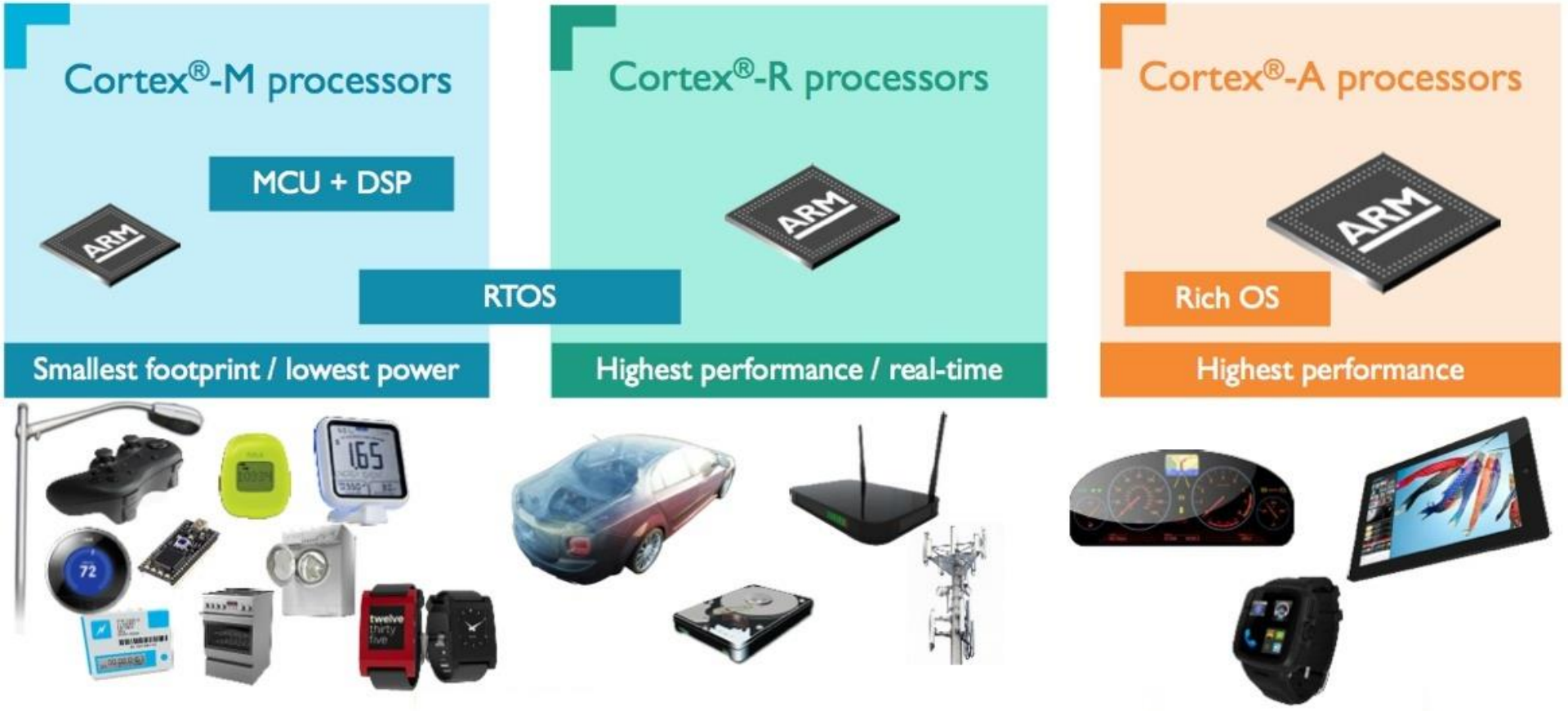
- Todas las soluciones ARM están diseñadas de tal forma que se maximice su rendimiento, reduciendo su consumo:
  - https://www.youtube.com/watch?v=7LqPJGnBPMM
- Tienen distintas versiones para distintos objetivos



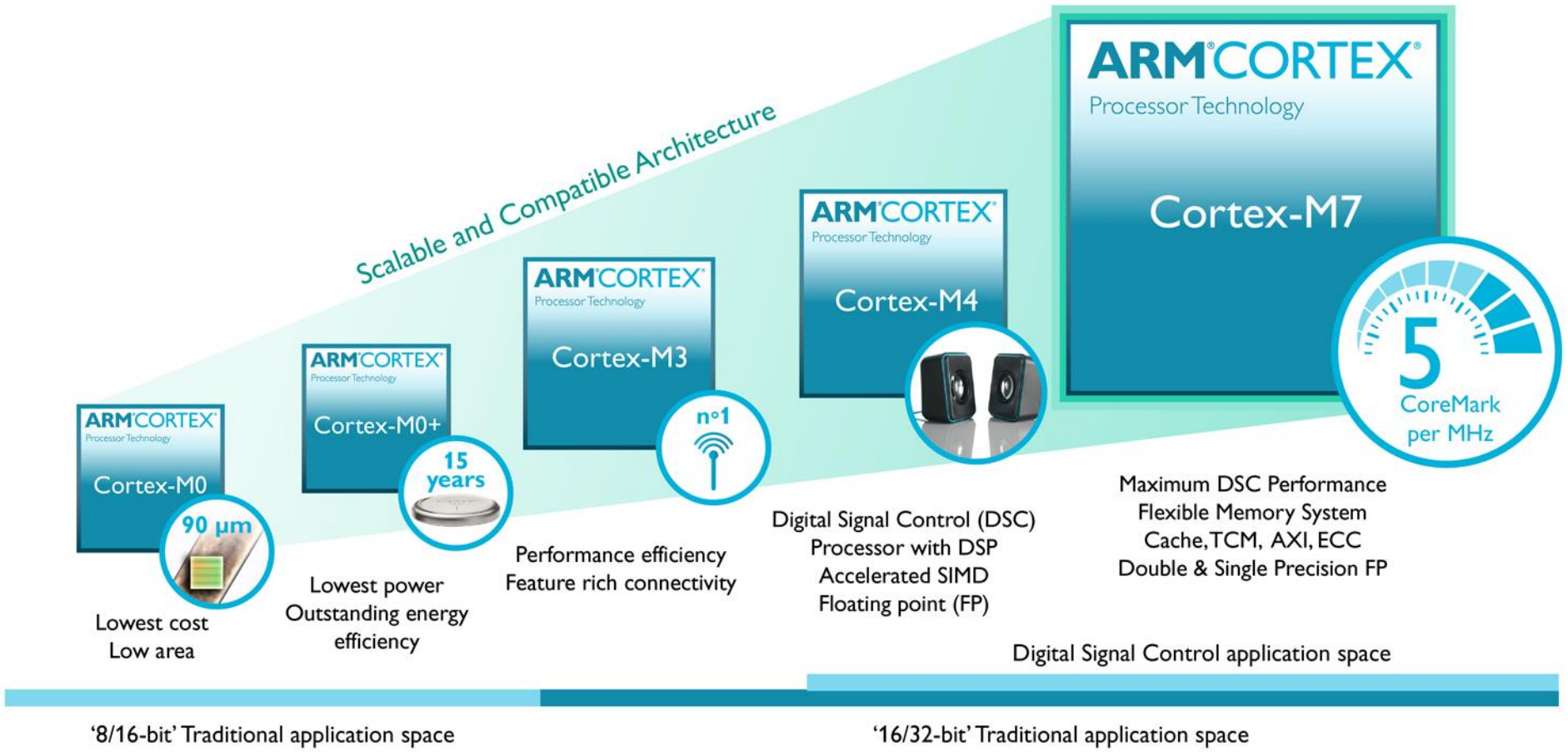
La


Classic
ARM Processors
Application
Cortex Processors
Embedded
Cortex Processors
Cortex-A15
Cortex-A9
ARM11MP
Cortex-A8
ARM926
ARM176JZ
Cortex-A7
SC300
SC000
SC100
ARM968
ARM1136J
Cortex-A5
Cortex-M3
Cortex-M1
ARM7TDMI
ARM946
ARM1156T2
Cortex-R4
Cortex-M4
Cortex-M0
ARMv4T
ARMv5TJ
ARMv6
ARMv7A/R
ARMv7M/ME
ARMv6M
ARM 32-Bit ISA
Thumb 16-Bit ISA
Thumb
Thumb
Thumb-2 Mixed ISA
Thumb-2
VFPv2
VFPv2
VFPv3
NVIC
NVIC
Jazelle
Jazelle
Jazelle
WIC
WIC
TrustZone
TrustZone
SIMD
SIMD
NEON
Virtualization

La Familia ARM



La Familia ARM



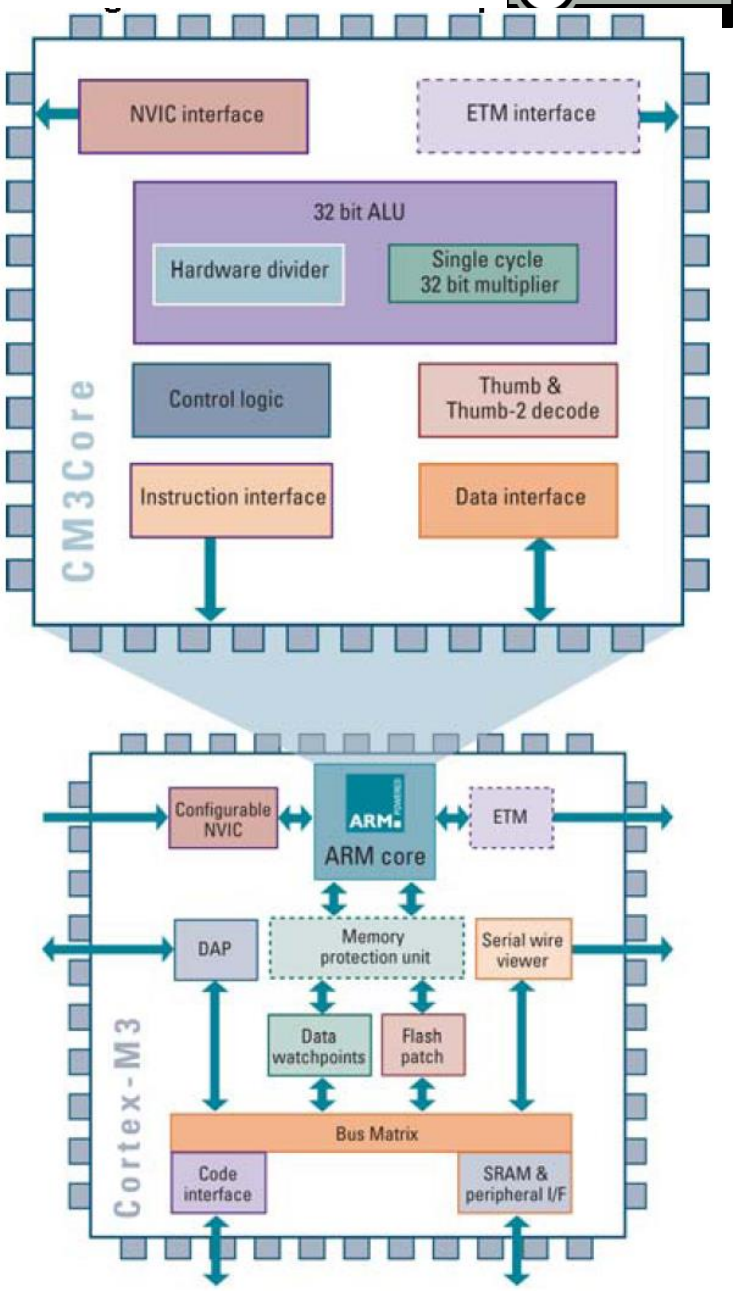
La Familia ARM

ARM Cortex-M instruction variations

ARM Core	Cortex M0[1]	Cortex M0+[2]	Cortex M1[3]	Cortex M3[4]	Cortex M4[5]	Cortex M7[6]	Cortex M23[7]	Cortex M33
Thumb-1 instructions	Most	Most	Most	Entire	Entire	Entire	Most	Entire
Thumb-2 instructions	Some	Some	Some	Entire	Entire	Entire	Some	Entire
Multiply instructions	32-bit result	32-bit result	32-bit result	32-bit result 64-bit result	32-bit result 64-bit result	32-bit result 64-bit result	32-bit result	32-bit result 64-bit result
Divide instructions	No	No	No	Yes	Yes	Yes	Yes	Yes
Saturated instructions	No	No	No	Some	Yes	Yes	No	Yes
DSP instructions	No	No	No	No	Yes	Yes	No	Optional
Floating-point instructions	No	No	No	No	Optional: SP	Optional: SP or SP & DP	No	Optional: SP
TrustZone instructions	No	No	No	No	No	No	Optional	Optional
Instruction pipeline	3 stages	2 stages	3 stages	3 stages	3 stages	6 stages	2 stages	3 stages
Computer architecture	Von Neuman	Von Neumann	Von Neumann	Harvard	Harvard	Harvard	Von Neumann	Harvard
ARM architecture	ARMv6-M[9]	ARMv6-M[9]	ARMv6-M[9]	ARMv7-M[10]	ARMv7E-M[10]	ARMv7E-M[10]	ARMv8-M[14]	ARMv8-M[14]

Arquitectura del Cortex-M3

- El ARM7 (CM3Core) es una Arquitectura Harvard
- A nivel de uso, el Cortex-M3 fusiona los dos mapas de memoria, convirtiendo el CM3Core en equivalente a una Arquitectura Von Neumann:
  - Bus direcciones de 32 bits
    - ***Se direccionan bytes***
  - Bus de datos de 32 bits
  - 23 registros
    - 13 de propósito general
    - 2 de puntero de pila (SP)
    - 1 PC
    - 1 registro de enlace (LR)
    - 1 registro de estado
    - 4 registros especiales



El Microcontrolador STM32L152

La Familia ARM de STMicroelectronics

Características Generales

- El micro STM32L15x (siendo la x una letra que define a implementaciones equivalentes del mismo microcontrolador) además del ARM Cortex-M3, incluye:
  - 128 KB de memoria Flash para programas
  - 16 KB de RAM estática
  - 4 KB de EEPROM para datos
  - Diversos periféricos integrados en el propio chip, entre ellos:
    - Pines I/O de propósito general tolerantes a 5V
    - Temporizadores de 32 bits (Timers) y uno de 24 (SysTick)
    - Conversor ADC de 12 bits
    - Conversor DAC de 12 bits
    - Controlador de Interrupciones Vectorizadas NVIC
    - Entradas de IRQ externa con disparo por nivel o flanco
    - Puertos Serie Asíncronos y Síncronos (USART, I^2C y SPI)
    - Reloj en Tiempo Real (RTC)
    - Varios canales de DMA
    - 7 modos de bajo consumo
    - Múltiples fuentes de reloj (internas y externas)
    - Comparador analógico
    - Circuito de Watch Dog ...

Bloques

LQFP64 10 x 10 mm

La familia de bajo consumo STM32L15xxx ofrece 3 encapsulados desde 48 a 100 patas, cada uno con diferentes periféricos

Diagrama de Bloques (detalle)

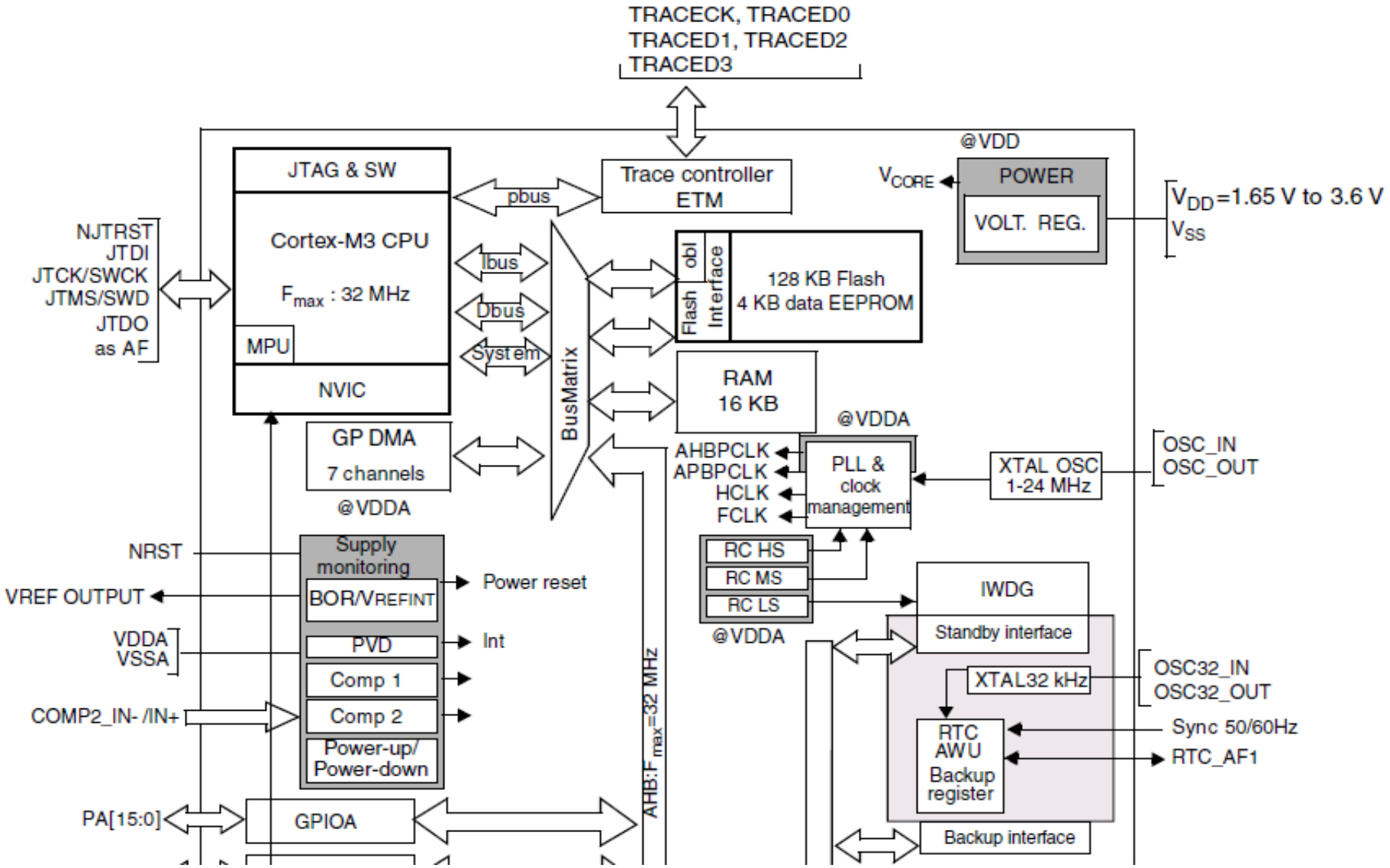
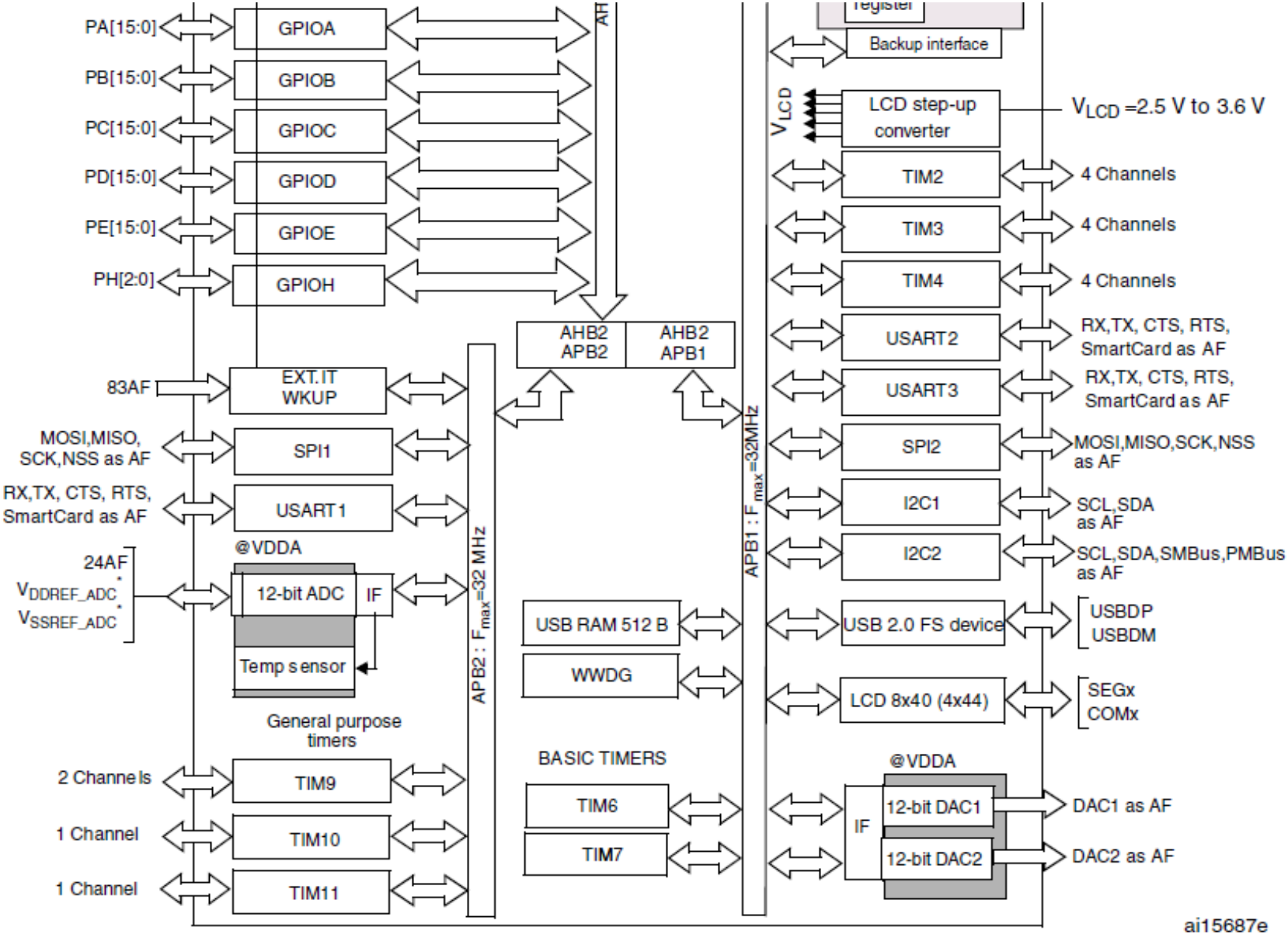
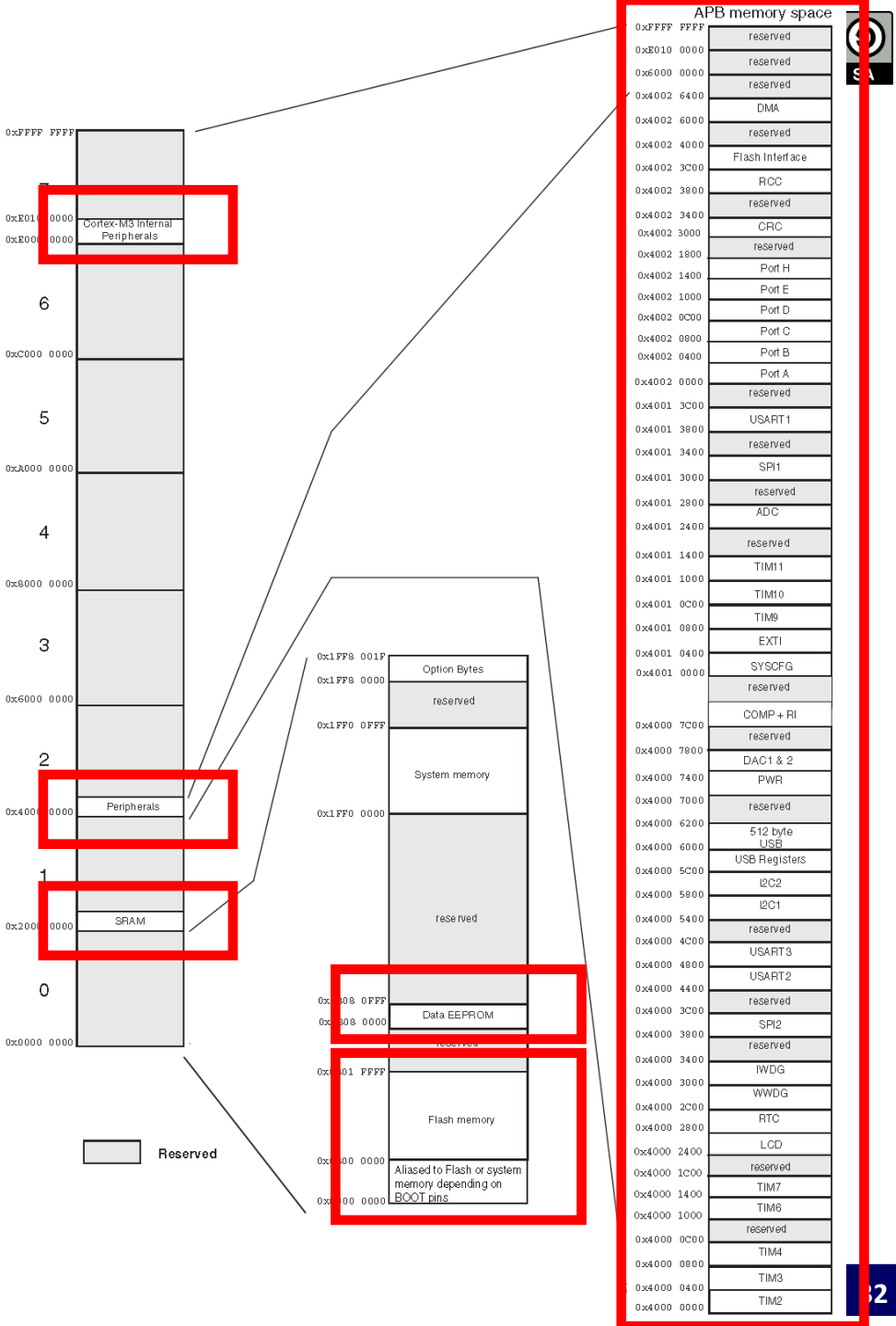


Diagrama de Bloques (detalle)



Mapa de Memoria

- Direccionamiento total de 4GB
- 128KB Flash
  - 0x0000 0000 – 0x0801 FFFF
- 16KB RAM estática (SRAM)
  - 0x2000 0000 – 0x2000 3FFF
- 4096B EEPROM
  - 0x0808 0000 – 0x0808 0FFF
- Periféricos del microcontrolador
  - 0x4000 0000 – 0x4002 63FF
- Periféricos del Cortex M3
  - 0xE000 0000 – 0xE010 FFFF

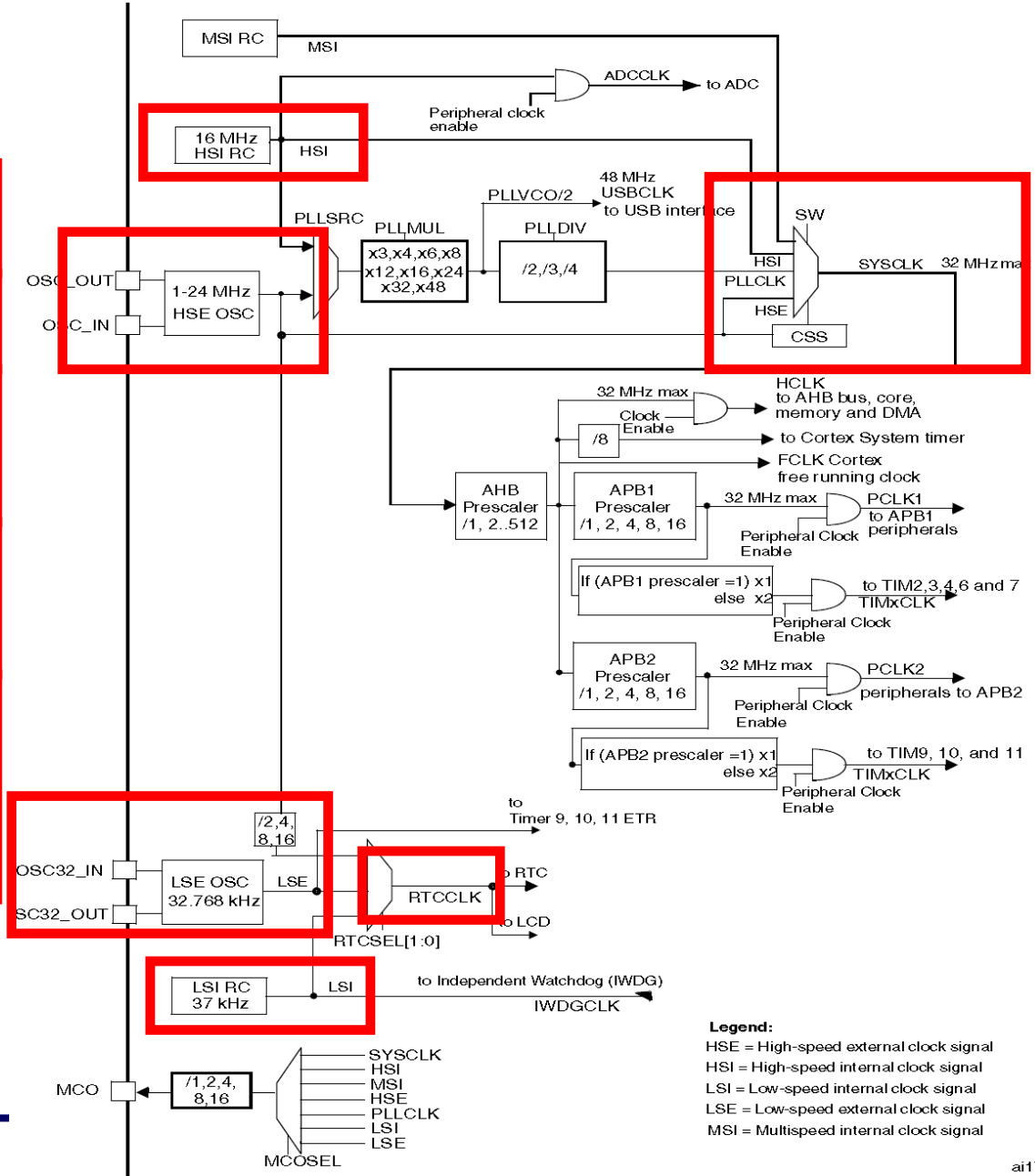


Los Periféricos vistos por la CPU

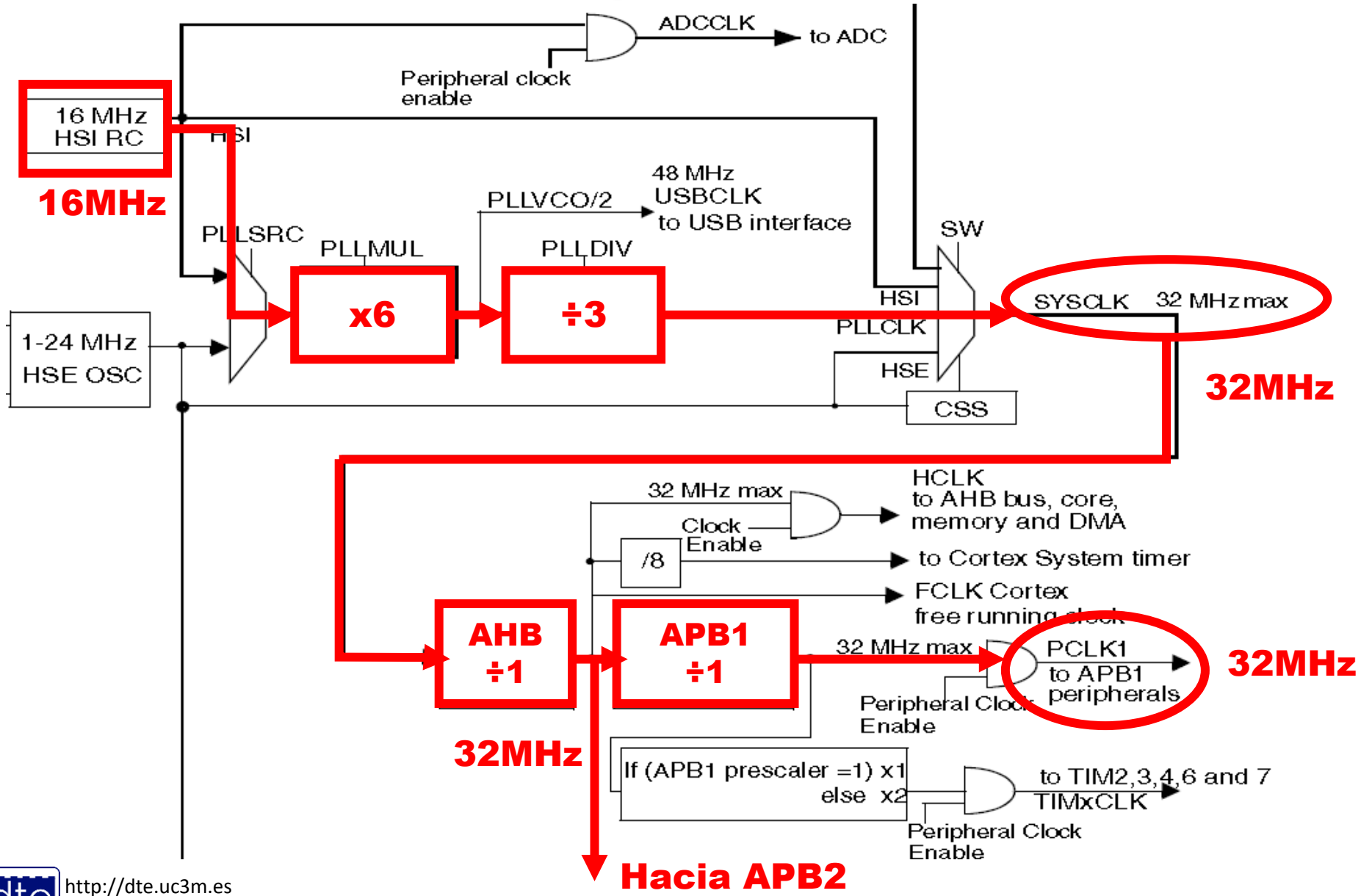
- Todo periférico, por complejo que sea, va a ser visto por la CPU como un conjunto de registros:
  - **De Datos:** los que van a contener los datos que se van a utilizar en el periférico y que se comunicarán a/desde la CPU
    - Generalmente serán de lectura y escritura
  - **De Estado:** los que van a contener información sobre el estado en el que se encuentra el periférico
    - Generalmente serán solo de lectura
  - **De Control:** lo que se van a escribir para configurar el periférico
    - Generalmente serán solo de escritura
- Para acceder a dichos registros, la CPU podrá hacerlo de dos formas:
  - Mediante instrucciones especiales de E/S
  - Como si fuese acceder a una dirección de memoria
    - ***Mapeado en Memoria de los Periféricos (lo normal)***

Subsistema de Reloj

- Diferentes fuentes de reloj:
  - HSI: Oscilador interno de alta velocidad (16MHz)
  - LSI: Oscilador interno de baja velocidad (37KHz)
  - HSE: Oscilador externo de alta velocidad (1 – 24 MHz)
  - LSE: Oscilador externo de baja velocidad (32,768 KHz)
- SYSCLK: Reloj del sistema
  - Puede proceder de HSI, de HSE, de MSI, o de un PLL procedente de HSI o HSE
- RTCCLK: Reloj en tiempo real del sistema



Subsistema de Reloj durante el curso



Nota Importante

Las capacidades del microcontrolador (y de cada uno de sus periféricos) son mucho mayores que las que se van a describir en este curso.

Esta reducción de capacidades se hace por motivos docentes, potenciando el aprendizaje de conceptos universales, y minorando el aprendizaje de conceptos específicos.

Notación RTL

Notación RTL

- El *Register Transfer Language* (RTL) es un lenguaje que simboliza la transferencia de datos entre registros
- Se utiliza, entre otras cosas, para representar el funcionamiento de un determinado proceso
  - La ejecución de una instrucción
  - El funcionamiento de la Unidad de Control
  - La secuencia de pasos que conlleva una interrupción
  - etc.
- Las variantes son muchas, por lo que se pueden encontrar especificaciones en RTL muy distintas de unos autores a otros
  - Mientras haya consistencia entre las representaciones, se puede considerar como válida

Notación RTL

- En este curso se va a utilizar una representación muy simplificada:
  - Paso del contenido de un registro a otro:
    - MAR $\Leftarrow$ PC ; lleva el contenido del PC al MAR
  - Paso del contenido de la memoria a un registro (operación de lectura)
    - MBR $\Leftarrow$ (MAR) ; lleva el contenido que tiene la memoria en la posición indicada por MAR, al registro MBR
  - Paso del contenido de un registro a memoria (operación de escritura)
    - (MAR) $\Leftarrow$ MBR ; lleva el contenido del registro MBR a la memoria, a la posición indicada por MAR
  - Operaciones con el operando fuente
    - PC $\Leftarrow$ PC + 1 ; incrementa el PC en una unidad

La Unidad de Control

La Unidad de Control

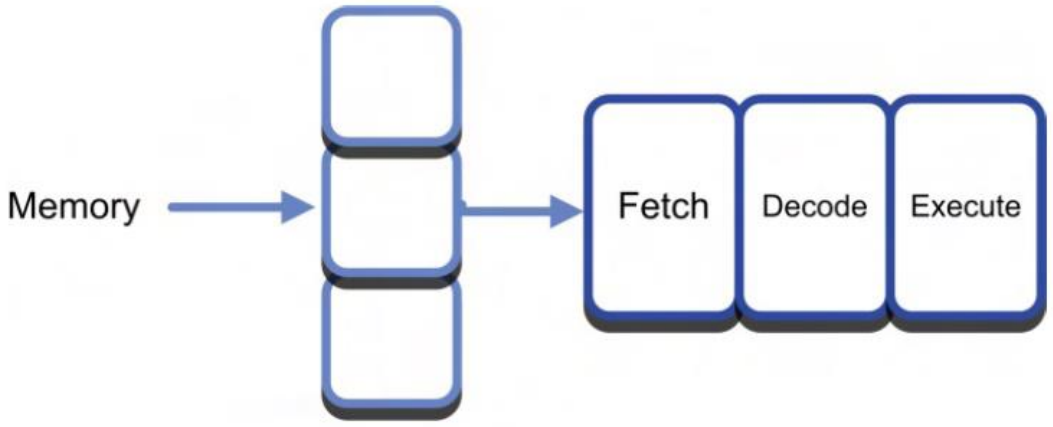
- Es el elemento que se encarga de gestionar el funcionamiento de toda la CPU
- Se trata de un autómata que se basa en ejecutar continuamente tres fases:
  - ***Fetch*** (captura): Se captura la instrucción de la memoria. Conlleva las siguientes operaciones:
    - MAR ⇐ PC
    - PC ⇐ PC + 1
    - MBR ⇐ (MAR) ; Lectura de memoria
    - IR ⇐ MBR
  - ***Decode*** (decodificación): Se interpreta el contenido del IR y se prepara a la CPU para la siguiente fase
  - ***Execute*** (ejecución): Se ejecutan los pasos necesarios para la consecución de la operación

La Unidad de Control

- Al tratarse de un autómata, funcionará a un ritmo marcado por el reloj del sistema:
  - En realidad se habla de ***ciclos máquina***
    - Dependiendo de la CPU, cada ciclo máquina puede ser un ciclo de reloj, o puede ser un número fijo de ciclos de reloj
  - Cada instrucción, en la ejecución de sus tres fases, consumirá un determinado número de ciclos máquina
    - La fase de fetch es común a todas
    - El número de ciclos en las otras dos fases puede ser distinto para cada instrucción
  - Cuanto mayor frecuencia pueda tener el reloj del sistema, más rápida será la ejecución del programa
  - Es importante tener en cuenta que la ejecución de una instrucción conlleva un consumo de tiempo
- La Unidad de Control genera tantas señales internas como necesite para controlar cada uno de los componentes de la CPU (**bus de control interno**)

La Unidad de Control

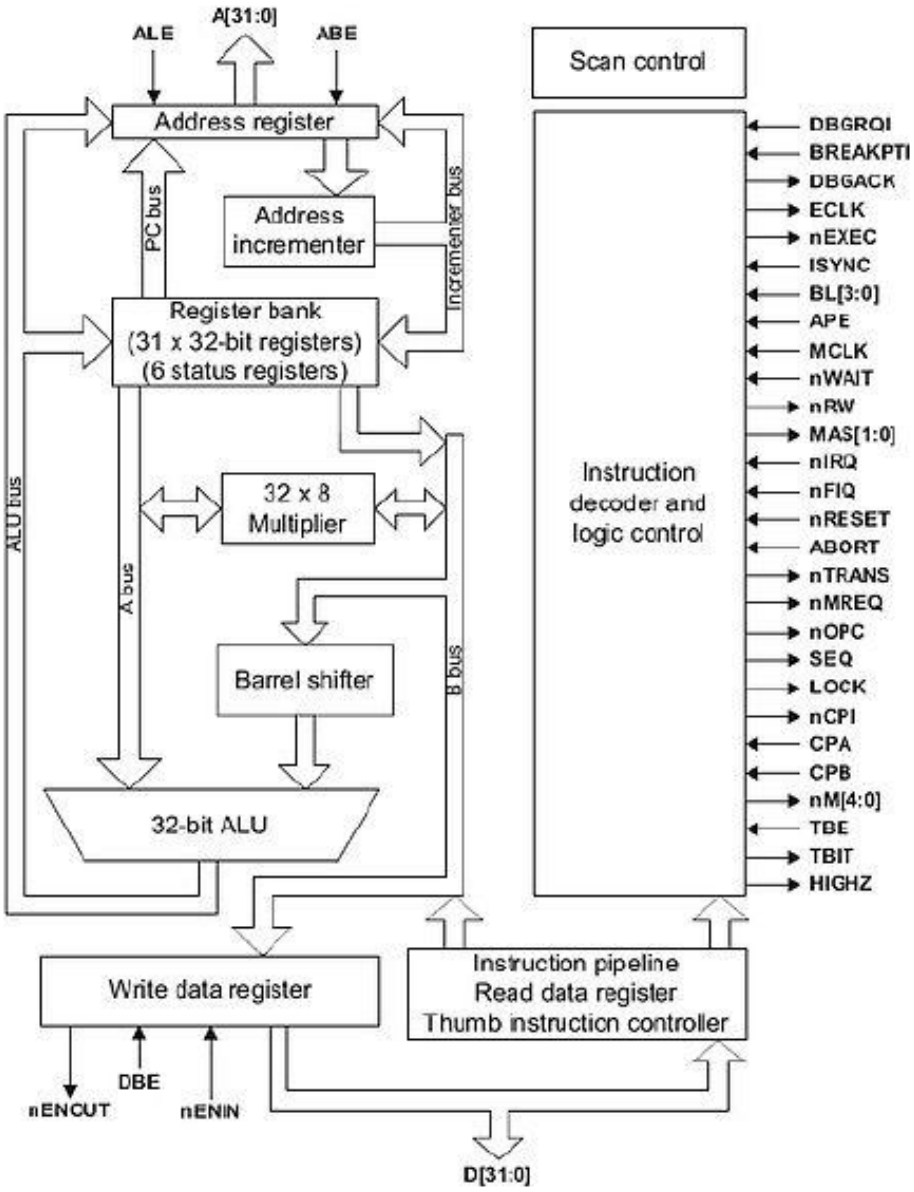
- Dependiendo de la Arquitectura de la CPU, el diseñador de la misma puede haber hecho una optimización de recursos internos, para procesar varias instrucciones en “paralelo”:
  - Mientras se está ejecutando una instrucción, se puede estar capturando la siguiente (pre-fetch)
  - A esta técnica se le denomina ***segmentación*** y el número de etapas (segmentos) puede variar de una a otra
  - El Cortex-M3 presenta una segmentación a 3 niveles:
    - Fetch
    - Decodificación
    - Ejecución



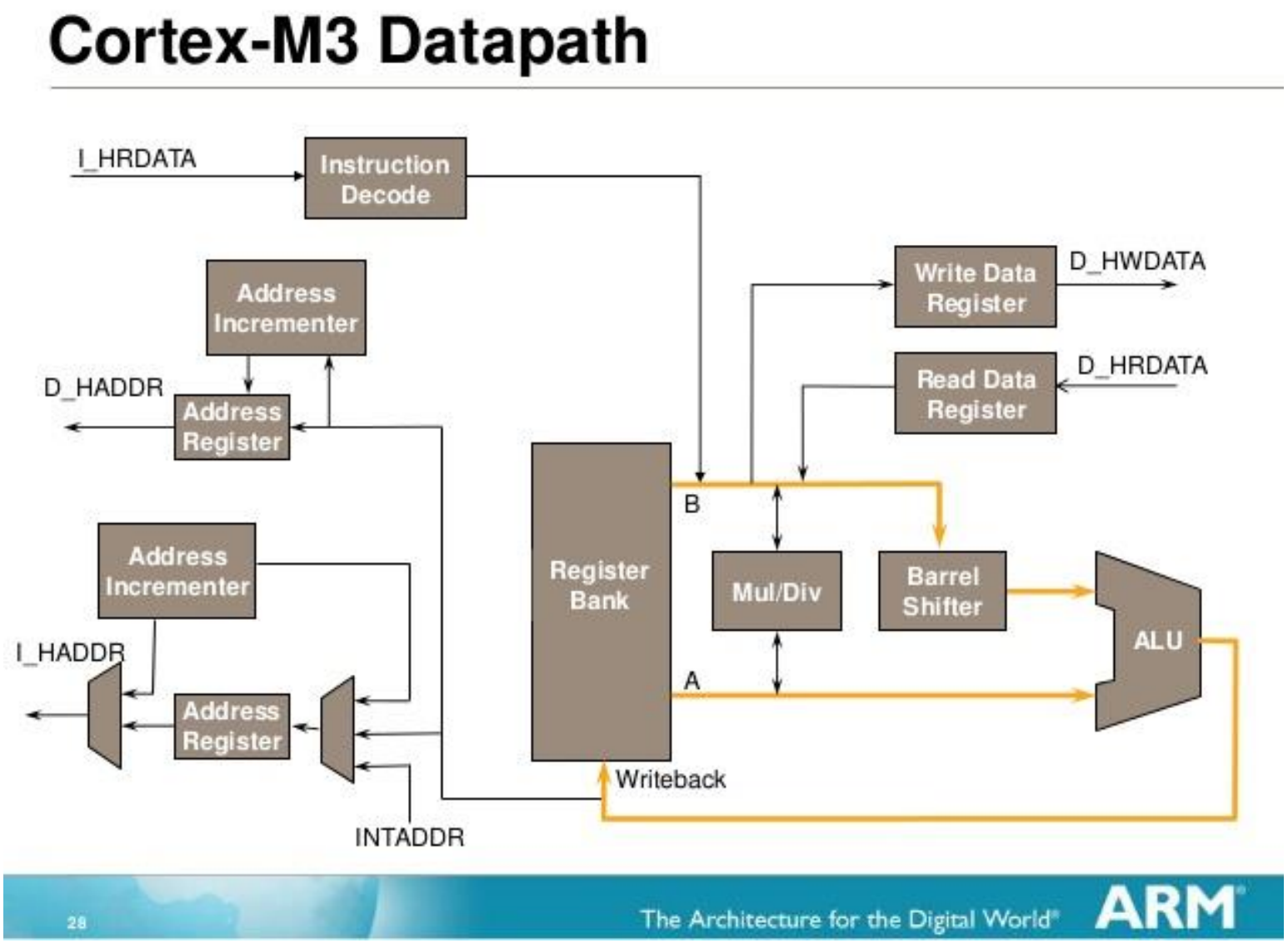
3 Stage Prefetch

La Unidad Aritmético Lógica y la Ruta de Datos

La Ruta de Datos del ARM7

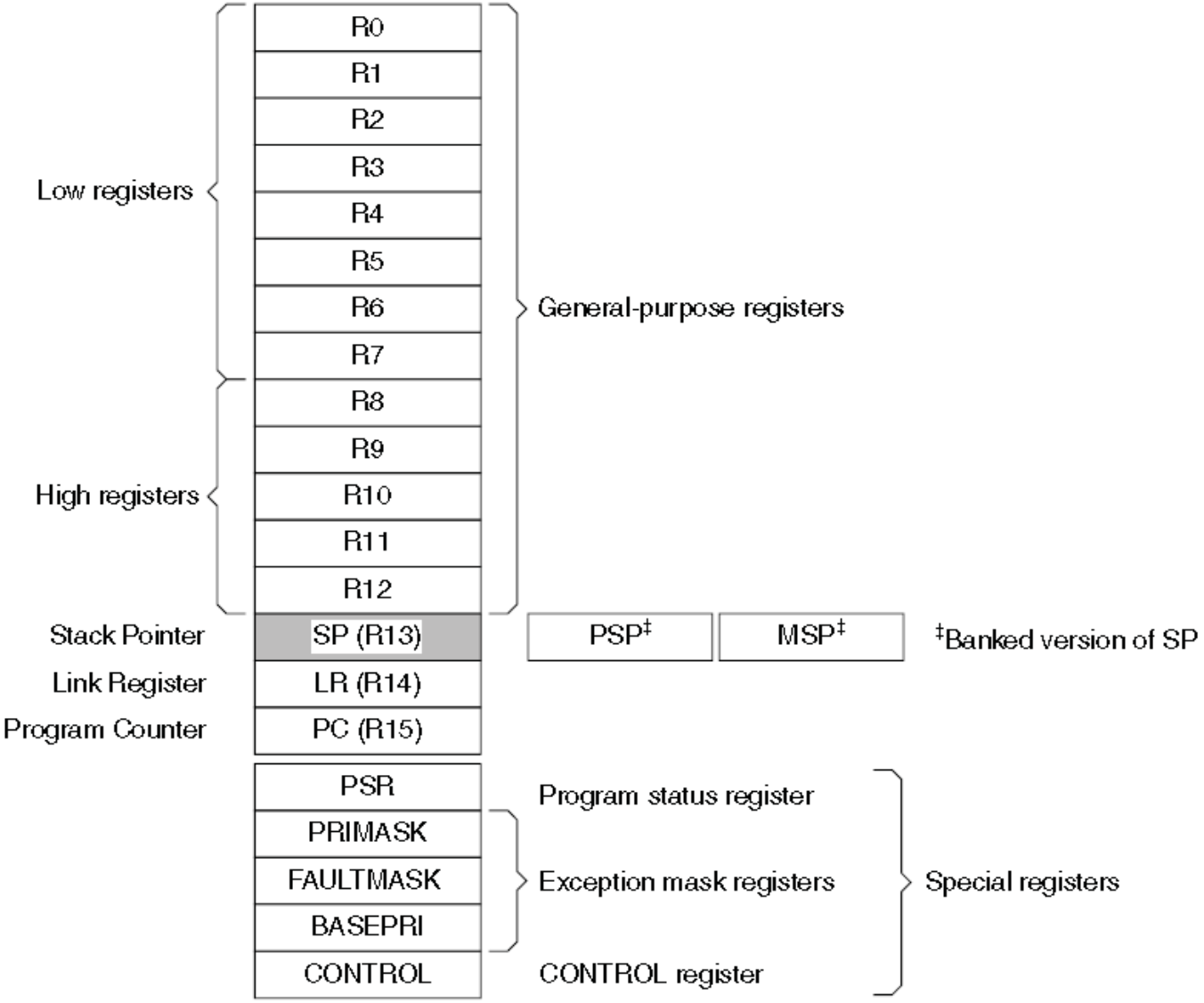


La Ruta de Datos del Cortex-M3



Los Registros de la CPU

Registros Internos del Cortex-M3



El Contador de Programa

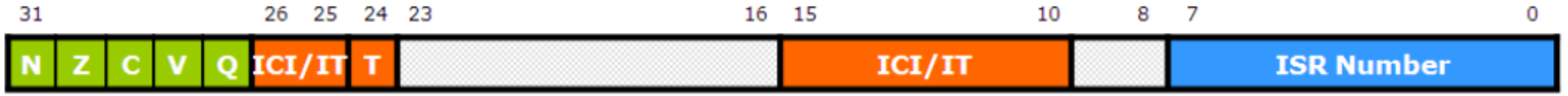
- Contiene en cada instante la dirección de la **siguiente instrucción** a capturar
- Se trata de un registro con las siguientes características:
  - Mismo tamaño que el bus de direcciones
  - Capacidad de ser modificado completamente por determinadas instrucciones
  - Capacidades especiales de "autoincremento" para no saturar la ALU en cada fase de *fetch* y captura de parámetros de la instrucción (caso de más de una palabra por instrucción)

El Registro de Estado (SR)

- Sirve para almacenar el estado en el que se encuentra la CPU tras la ejecución de una instrucción
- El estado se indica con una serie de flags, que son cada uno de los bits de ese registro de estado
- Los flags típicos son:
  - Z: se activa si la última operación ha dado como resultado un 0
  - N: se activa si la última operación ha dado como resultado un número negativo
  - C: se activa si la última operación ha producido acarreo
  - V: se activa si la última operación ha dado desbordamiento
- También puede haber otros flags que indiquen interrupción, etc. (o que habiliten esos eventos)

El Registro de Estado (SR)

- En el Cortex-M3 se tienen 3 registros de estado, que se combinan en el PSR:
  - Application Program Status Register (APSR)
    - Que contiene los flags de estado
  - Interrupt Program Status Register (IPSR)
    - Que contiene información respecto a la RAI en ejecución
  - Execution Program Status Register (EPSR)
    - Que contiene información de ejecución del programa



El Puntero de Pila (SP)

- En las CPUs actuales, es común disponer de un registro adicional, denominado Puntero de Pila (SP)
  - Es un registro que contiene una dirección de memoria RAM
  - Esa dirección se incrementa o decrementa según se hagan operaciones con “la pila”
- La Pila:
  - Es una zona de memoria dedicada a almacenar información en formato LIFO (el último que entra es el primero que sale)
  - Se implementa en memoria RAM, y puede crecer hacia direcciones altas o hacia direcciones bajas
  - El SP puede apuntar a la última dirección escrita, o a la primera libre
    - Por ejemplo, si crece hacia direcciones más bajas, y apunta a la primera dirección libre, una operación de escritura en la pila (PUSH) sería:
      - (SP) $\Leftarrow$ MBR ; SP $\Leftarrow$ SP-1
    - Una de lectura (PULL) sería
      - SP $\Leftarrow$ SP+1 ; MBR $\Leftarrow$ (SP)

El Puntero de Pila (SP)

- El Cortex-M3 utiliza una pila completa descendente (el SP apunta al último elemento introducido en la pila; cuando se introduce un nuevo elemento en la pila, se decrementa el SP y luego se introduce)
- El Cortex-M3 implementa 2 pilas, la *main stack* y la *process stack*:
  - Cada una tiene su propio SP
  - Cuando se está gestionando una excepción, sólo se utiliza la *main stack*

La Memoria Principal

La Memoria Principal

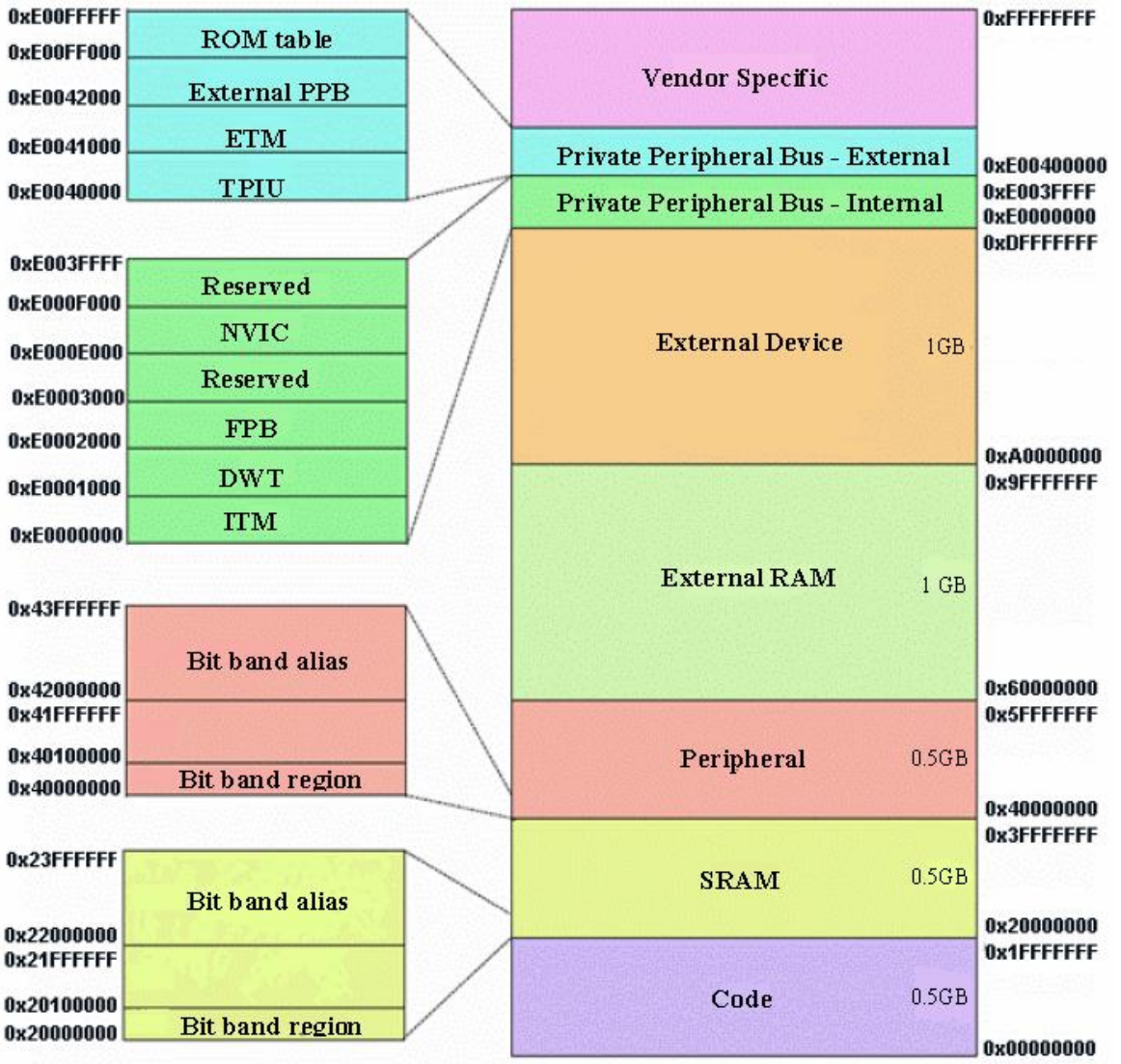
- Manteniendo la filosofía de Von Neumann, la memoria almacena tanto datos como instrucciones
  - Las instrucciones deberían estar en memoria no-volátil (ROM, EPROM, EEPROM, etc.)
  - Los datos deberían estar en memoria re-escribible fácilmente (RAM)
- Normalmente se utilizan distintos chips de memoria y se guardan instrucciones y datos en distintas partes del **mapa de memoria:**
  - Distintos chips que comparten líneas de datos y de direcciones
    - Cada chip de una tecnología, según necesidades
  - Selección del chip al que se accede dependiendo del rango al que pertenece la dirección (**decodificador de direcciones**)

La Memoria Principal

- Bancos de Memoria:
  - Se puede ampliar memoria creando artificialmente bancos de memoria
    - Una determinada posición de memoria (un registro) almacena el número de banco con el que se está trabajando
    - Al acceder a memoria, en decodificador de direcciones toma como una de sus entradas también el valor de esa posición, y decide que chip selecciona
    - Para cambiar de banco sólo hay que volver a escribir en memoria el número del nuevo banco de memoria al que acceder
  - De esta forma se obtiene una capacidad de 2^AxB
    - A es el tamaño del bus de direcciones
    - B el número de bancos que se contemplan

La Memoria Principal

- El Cortex-M3 describe un modelo de memoria como el de la figura
- Deja espacio para que cada vendedor mapee aquellos recursos que quiera ofrecer
  - Memoria
  - Periféricos
- Se direccionan bytes (no palabras de 32 bits)



Instrucciones

El Registro de Instrucción (IR)

- Es el encargado de mantener la instrucción que se va a ejecutar, para que se decodifique y se emprendan las acciones necesarias
- El tamaño del IR determina el número máximo de instrucciones, por las combinaciones posibles (8 bits = 256 operaciones máximas)
- Sin embargo, las instrucciones se codifican de forma muy estructurada, lo que limita las posibilidades y simplifica la decodificación de las mismas, y el trabajo del programador
  - Como mínimo, las instrucciones se codifican en dos partes:
    - ***Opcode*** o código de operación: es el código que indica la instrucción a ejecutar y su variante (modo de direccionamiento, etc.)
    - ***Parámetros***: normalmente determinan un (o varios) operando o la dirección de un (o varios) operando.

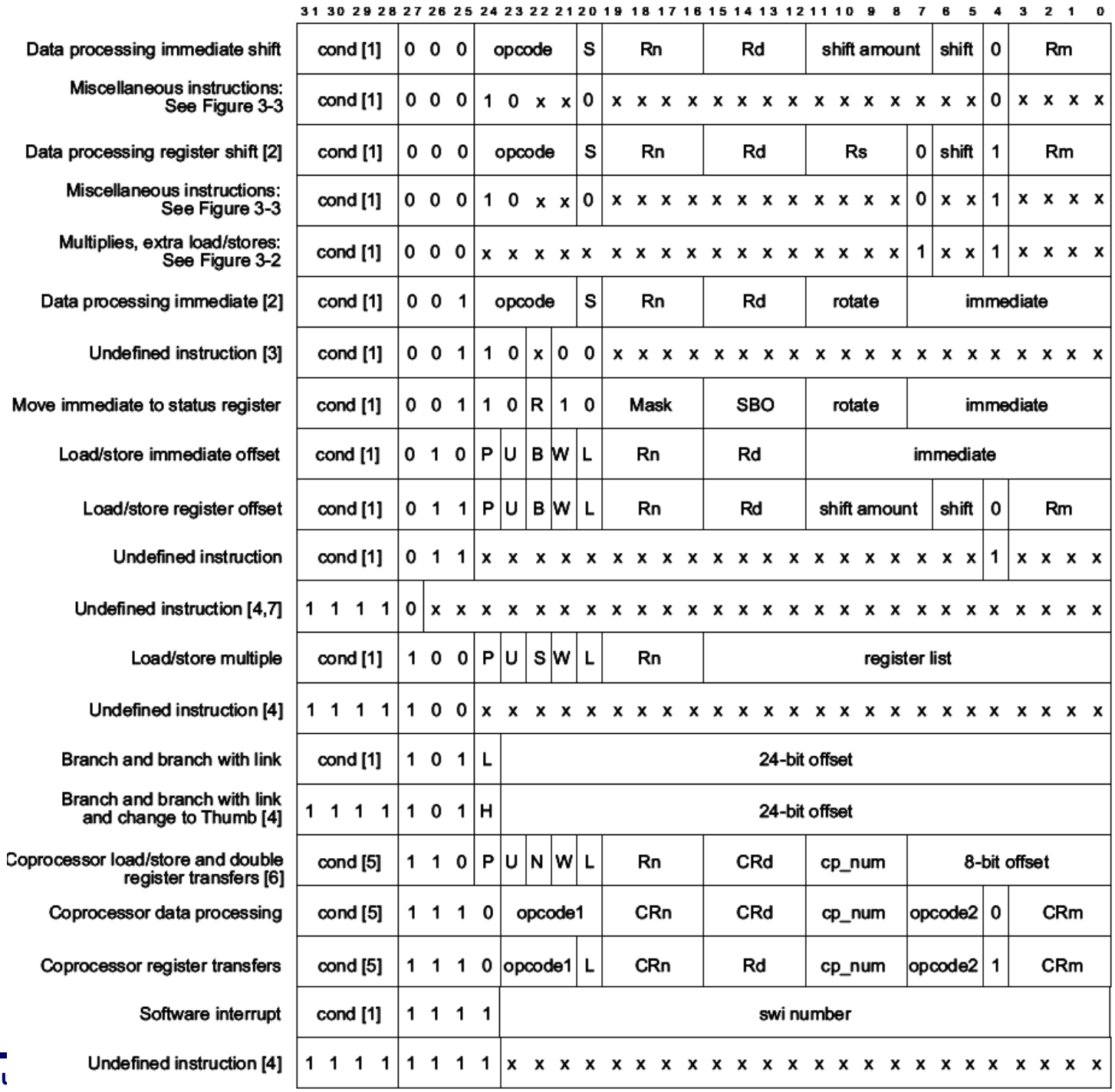
El Registro de Instrucción (IR)

- Respecto al número de instrucciones que contempla una CPU, existen las siguientes arquitecturas:
  - **CISC:** Son CPUs que contemplan un gran número de instrucciones
    - Normalmente son instrucciones complejas, con gran número de variantes
    - Muchas de ellas se utilizan un número muy limitado de veces
    - Facilitan mucho la programación, al contemplar operaciones complejas
  - **RISC:** Un número de instrucciones reducido
    - Instrucciones muy sencillas, normalmente de una única palabra de memoria
    - La CPU es mucho más sencilla (pequeña y barata)
    - La programación se complica, al tener que hacer operaciones no excesivamente complejas, con un gran número de instrucciones sencillas
    - El ARM7TDMI presenta una arquitectura RISC

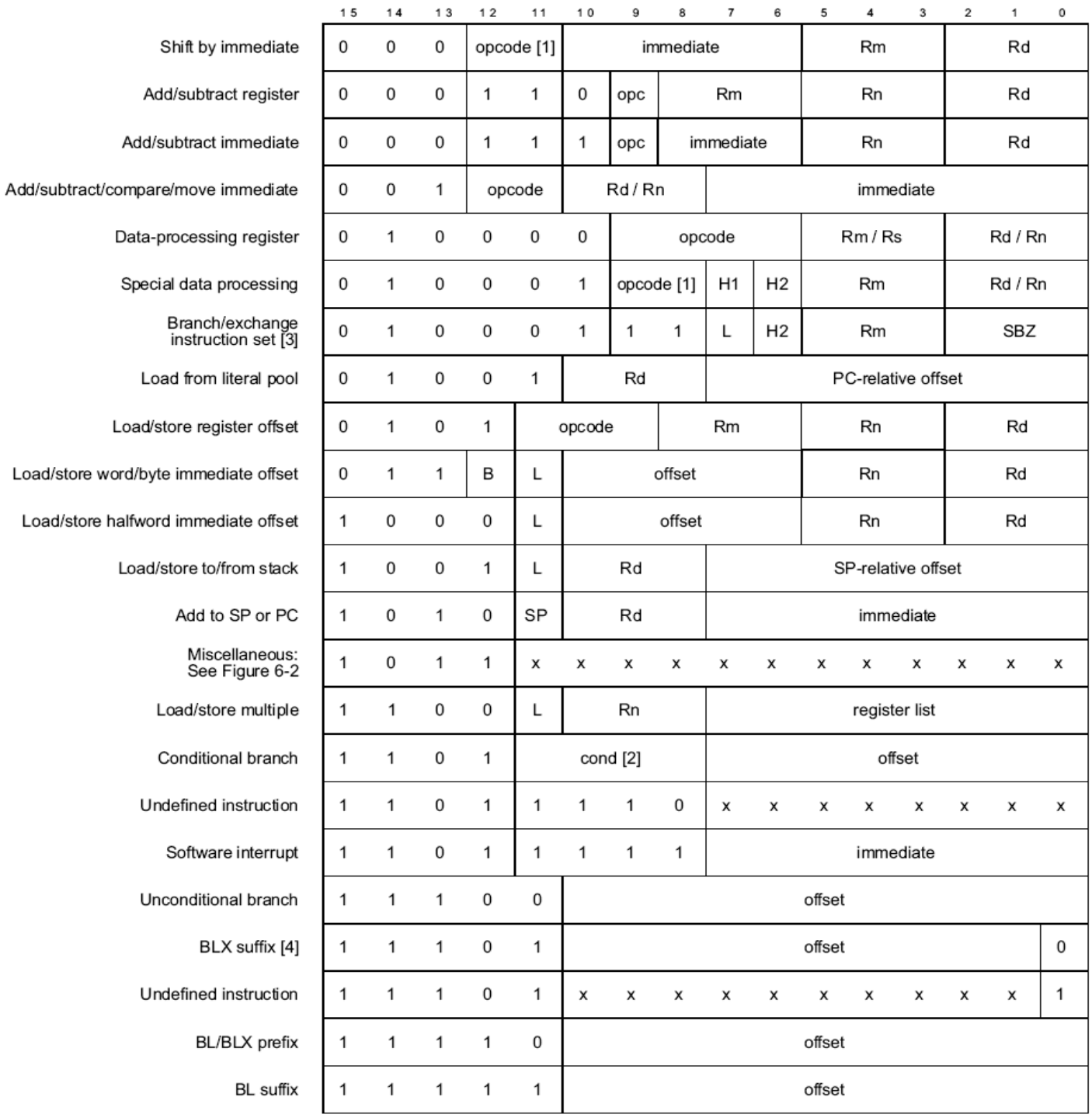
El Registro de Instrucción (IR)

- El IR suele tener el tamaño de una palabra de memoria.
  - Pero las instrucciones pueden ocupar una o varias palabras
    - De ocupar más de una palabra, el opcode se encuentra en la primera
- Las instrucciones del Cortex-M3 tiene las siguientes características:
  - Existen 2 juegos de instrucciones: Thumb y Thumb-2
    - Aquí se verá solo el Thumb-2
  - Todas las instrucciones son de 32 bits (1 palabra)
    - Aunque algunas de las instrucciones se pueden codificar en 16 bits
  - Arquitectura *load-store*
    - Todas las operaciones trabajan sólo con registros y literales, salvo las de transferencia de datos
  - Contemplan 3 operandos (dos fuentes y uno destino)
  - Ejecución condicional de todas las instrucciones

ARM 7

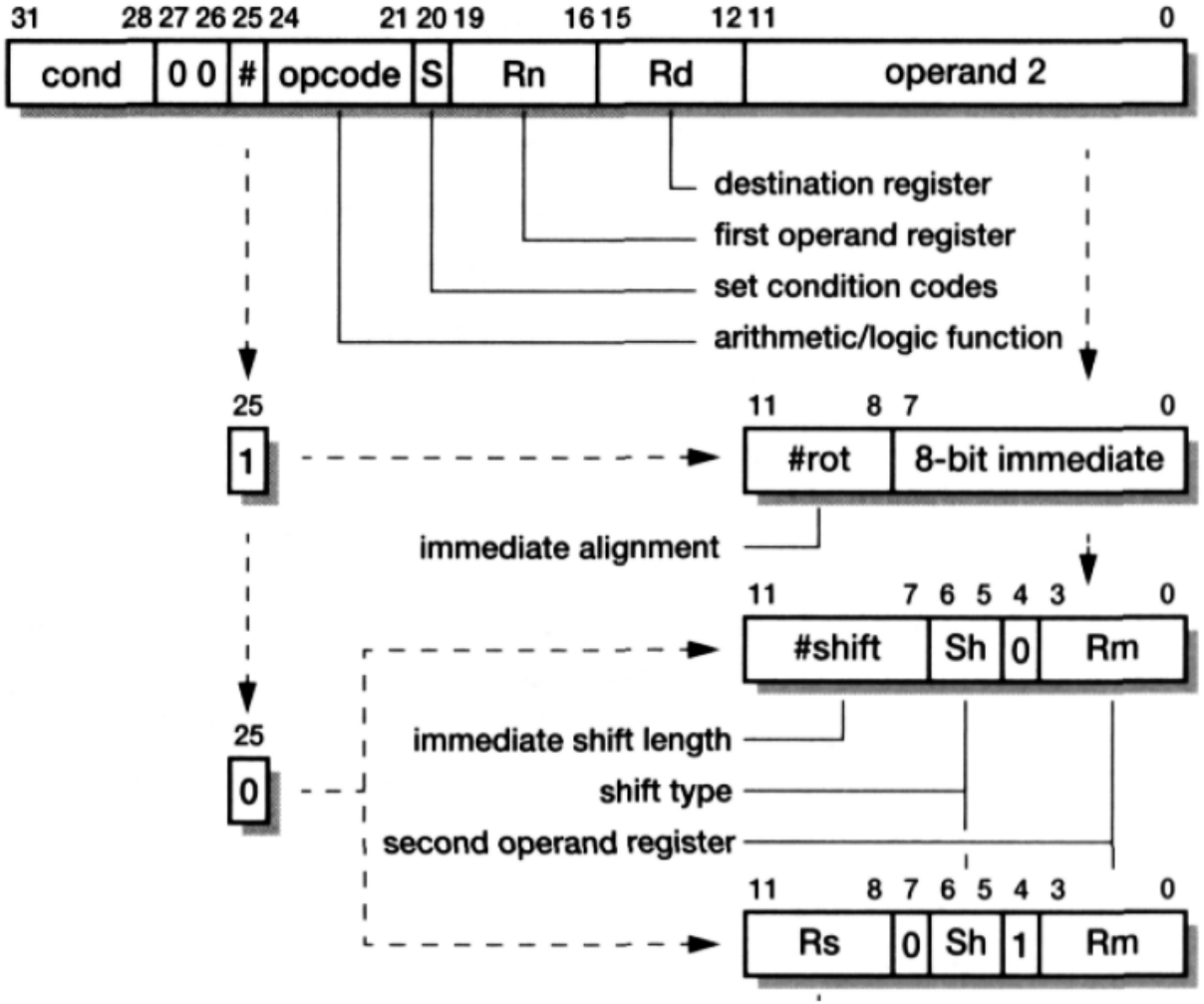

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Data processing immediate shift
cond [1] 0 0 0 opcode S Rn Rd shift amount shift 0 Rm
Miscellaneous instructions: See Figure 3-3
cond [1] 0 0 0 1 0 x x 0 x x x x x x x x x x x x 0 x x x x
Data processing register shift [2]
cond [1] 0 0 0 opcode S Rn Rd Rs 0 shift 1 Rm
Miscellaneous instructions: See Figure 3-3
cond [1] 0 0 0 1 0 x x 0 x x x x x x x x x x x x 0 x x 1 x x x x
Multiplies, extra load/stores: See Figure 3-2
cond [1] 0 0 0 x x x x x x x x x x x x x x x x x x 1 x x 1 x x x x
Data processing immediate [2]
cond [1] 0 0 1 opcode S Rn Rd rotate immediate
Undefined instruction [3]
cond [1] 0 0 1 1 0 x 0 0 x x x x x x x x x x x x x x x x x x x x
Move immediate to status register
cond [1] 0 0 1 1 0 R 1 0 Mask SBO rotate immediate
Load/store immediate offset
cond [1] 0 1 0 P U B W L Rn Rd immediate
Load/store register offset
cond [1] 0 1 1 P U B W L Rn Rd shift amount shift 0 Rm
Undefined instruction
cond [1] 0 1 1 x x x x x x x x x x x x x x x x x x x x 1 x x x x
Undefined instruction [4,7]
1 1 1 1 0 x x x x x x x x x x x x x x x x x x x x x x x x x x x
Load/store multiple
cond [1] 1 0 0 P U S W L Rn register list
Undefined instruction [4]
1 1 1 1 1 0 0 x x x x x x x x x x x x x x x x x x x x x x x x
Branch and branch with link
cond [1] 1 0 1 L 24-bit offset
Branch and branch with link and change to Thumb [4]
1 1 1 1 1 0 1 H 24-bit offset
Coprocessor load/store and double register transfers [6]
cond [5] 1 1 0 P U N W L Rn CRd cp_num 8-bit offset
Coprocessor data processing
cond [5] 1 1 1 0 opcode1 CRn CRd cp_num opcode2 0 CRm
Coprocessor register transfers
cond [5] 1 1 1 0 opcode1 L CRn Rd cp_num opcode2 1 CRm
Software interrupt
cond [1] 1 1 1 1 swi number
Undefined instruction [4]
1 1 1 1 1 1 1 1 x x x x x x x x x x x x x x x x x x x x x x x x

Thumb16


15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Shift by immediate
0 0 0 opcode [1] immediate Rm Rd
Add/subtract register
0 0 0 1 1 0 opc Rm Rn Rd
Add/subtract immediate
0 0 0 1 1 1 opc immediate Rn Rd
Add/subtract/compare/move immediate
0 0 1 opcode Rd / Rn immediate
Data-processing register
0 1 0 0 0 0 opcode Rm / Rs Rd / Rn
Special data processing
0 1 0 0 0 1 opcode [1] H1 H2 Rm Rd / Rn
Branch/exchange instruction set [3]
0 1 0 0 0 1 1 1 L H2 Rm SBZ
Load from literal pool
0 1 0 0 1 Rd PC-relative offset
Load/store register offset
0 1 0 1 opcode Rm Rn Rd
Load/store word/byte immediate offset
0 1 1 B L offset Rn Rd
Load/store halfword immediate offset
1 0 0 0 L offset Rn Rd
Load/store to/from stack
1 0 0 1 L Rd SP-relative offset
Add to SP or PC
1 0 1 0 SP Rd immediate
Miscellaneous: See Figure 6-2
1 0 1 1 x x x x x x x x x x x x
Load/store multiple
1 1 0 0 L Rn register list
Conditional branch
1 1 0 1 cond [2] offset
Undefined instruction
1 1 0 1 1 1 1 0 x x x x x x x x
Software interrupt
1 1 0 1 1 1 1 1 immediate
Unconditional branch
1 1 1 0 0 offset
BLX suffix [4]
1 1 1 0 1 offset 0
Undefined instruction
1 1 1 0 1 x x x x x x x x x x 1
BL/BLX prefix
1 1 1 1 0 offset
BL suffix
1 1 1 1 1 offset

El Registro de Instrucción (IR) – ARM7

- Ejemplo de instrucción genérica de Procesado de Datos:



El Registro de Instrucción (IR) – ARM7

- Ejemplo de instrucción genérica de Transferencia de Datos: