TEMA 11: ARQUITECTURA INTERNA DE UNA CPU

Sistemas Digitales basados en Microprocesador

Grado en Ingeniería Telemática

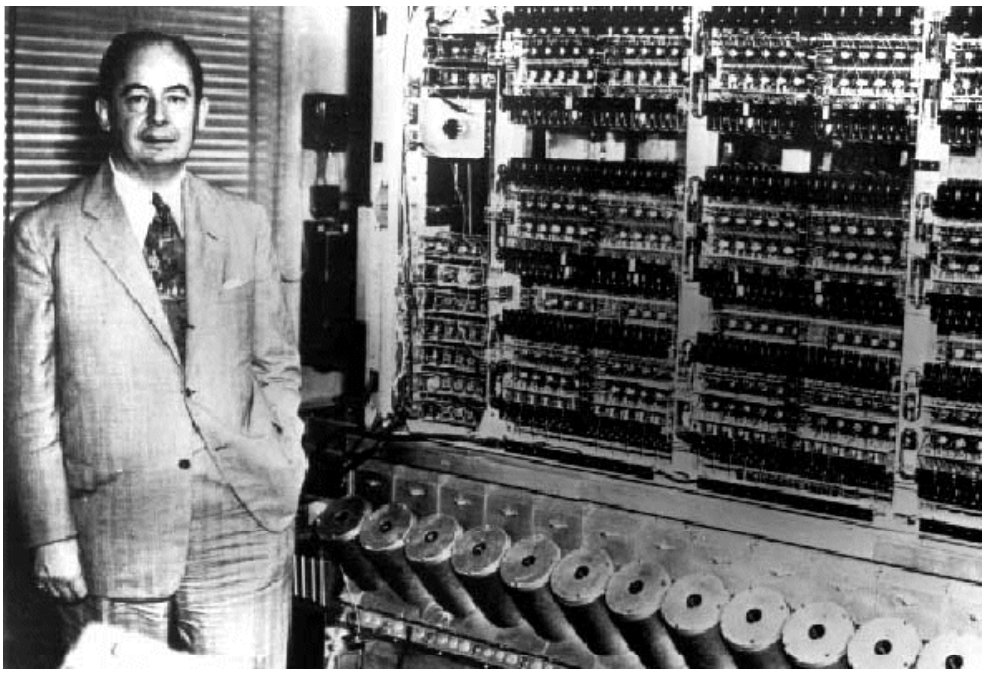
© Raúl Sánchez Reíllo

ÍNDICE

Arquitectura de Von Neumann

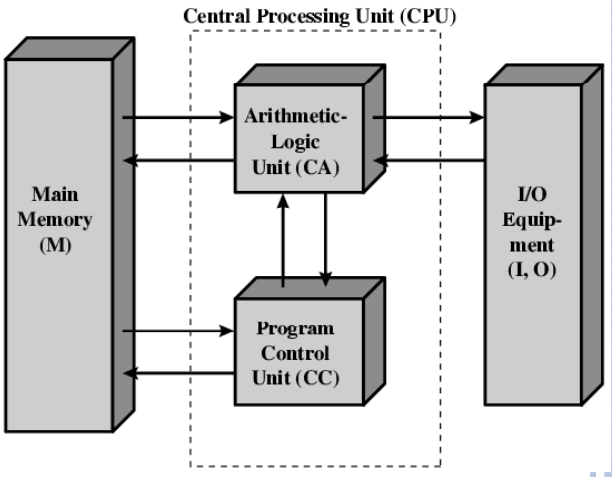
- John Von Neumann, en su artículo del año 1945, definió una computadora de propósito general, basada en la idea de **programa almacenado**
- Los componentes principales eran:
  - Una memoria principal
    - Almacenaba tanto datos como instrucciones
  - Una unidad de cálculo para operaciones aritméticas 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

Arquitectura de Von Neumann



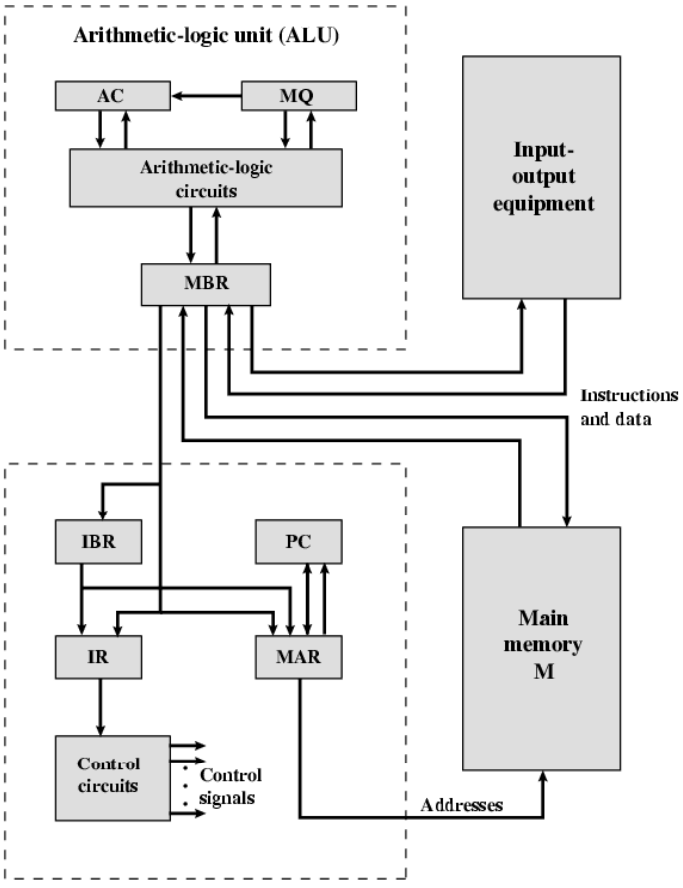
Arquitectura de Von Neumann

- 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



Arquitectura de Von Neumann

- Registros de la CPU:
  - MBR: Buffer de Memoria
  - MAR: Direccionamiento de Memoria
  - IR: Registro de Instrucción
  - IBR: Buffer del IR
  - PC: Contador de Programa
  - AC: Acumulador
  - MQ: Cociente Multiplicador
- La comunicación entre registros se hace mediante buses internos
  - Datos
  - Direcciones



Arquitectura de 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
  - Modificación de Direcciones
- 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 de Von Neumann

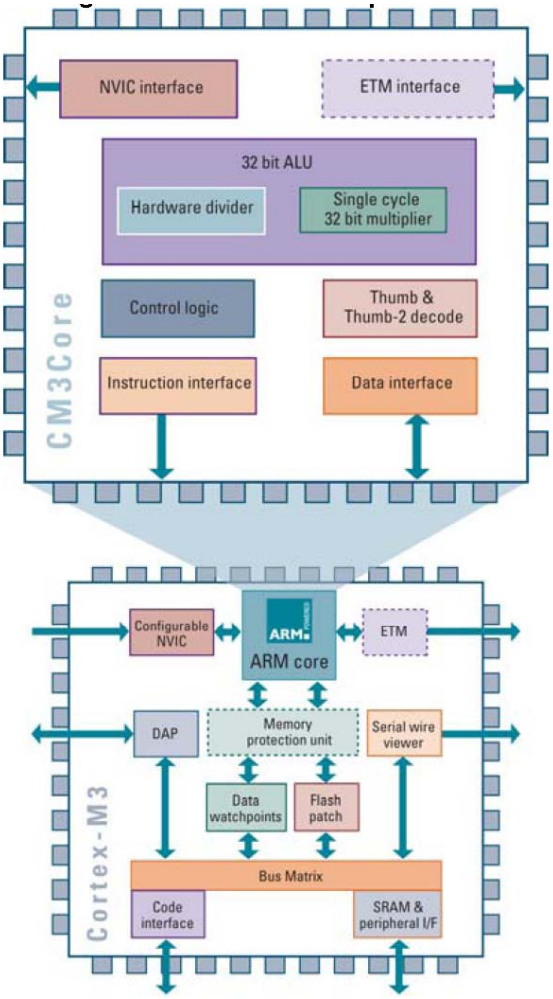
- Ampliaciones posteriores han dado lugar a dos tipos de arquitecturas:
  - Basada en Acumulador:
    - Es la original de Von Neumann (aunque a día de hoy pueden tener más de un acumulador)
    - Casi toda operación tiene como fuente o como destino el acumulador
  - Basada en Registros:
    - Surge para 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

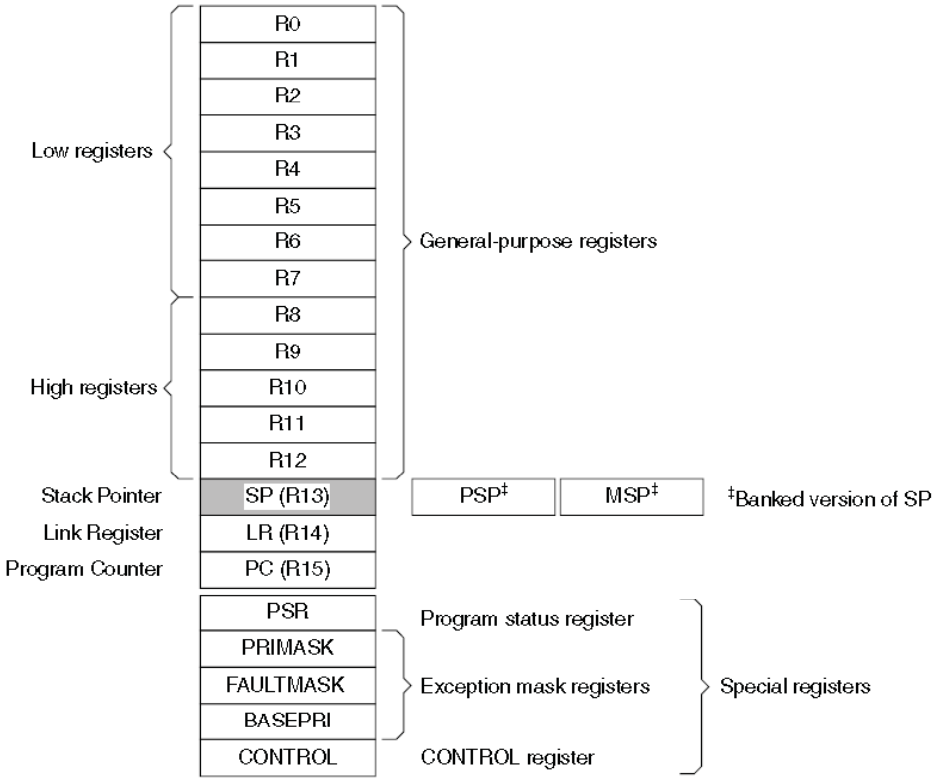
- Se elimina el concepto de Memoria Principal. En esta arquitectura existe:
  - Una memoria exclusivamente para datos
  - Una memoria exclusivamente para instrucciones
  - Buses (tanto de datos, como de direcciones) diferenciados para cada una de las memorias
    - Sus números de líneas pueden ser distintos
      - 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 del Cortex-M3

- Es una Arquitectura Harvard:
  - 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



Registros Internos del Cortex-M3



La Unidad de Control

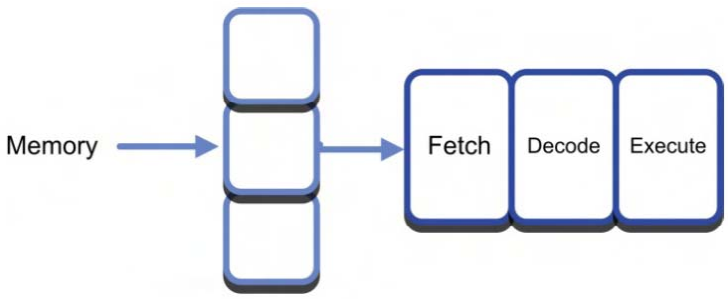
- Es el elemento que se encarga de gestionar el funcionamiento de toda la CPU
- Se trata de un autómata que se basa en ejecutar continuamente tres fases:
  - ***Fetch*** (captura): Se captura la instrucción de la memoria. Conlleva las siguientes operaciones:
    - MAR ⇐ PC
    - PC ⇐ PC + 1
    - MBR ⇐ (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

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.

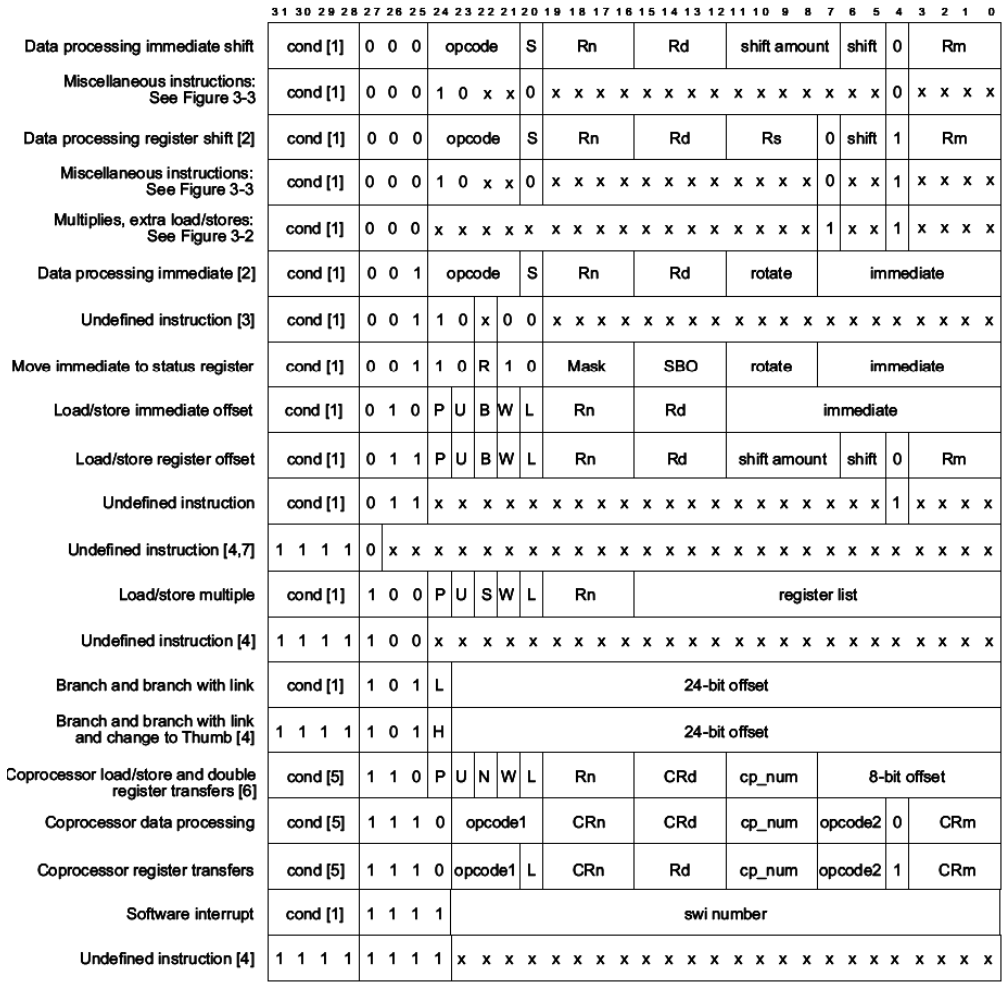
El Registro de Instrucción (IR)

- 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

ARM7

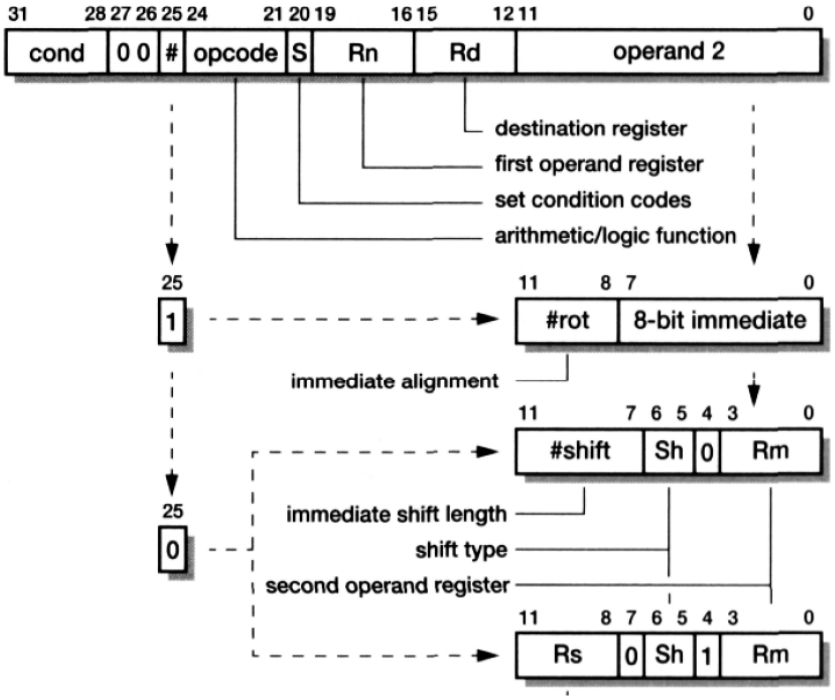


THUMB16

	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										

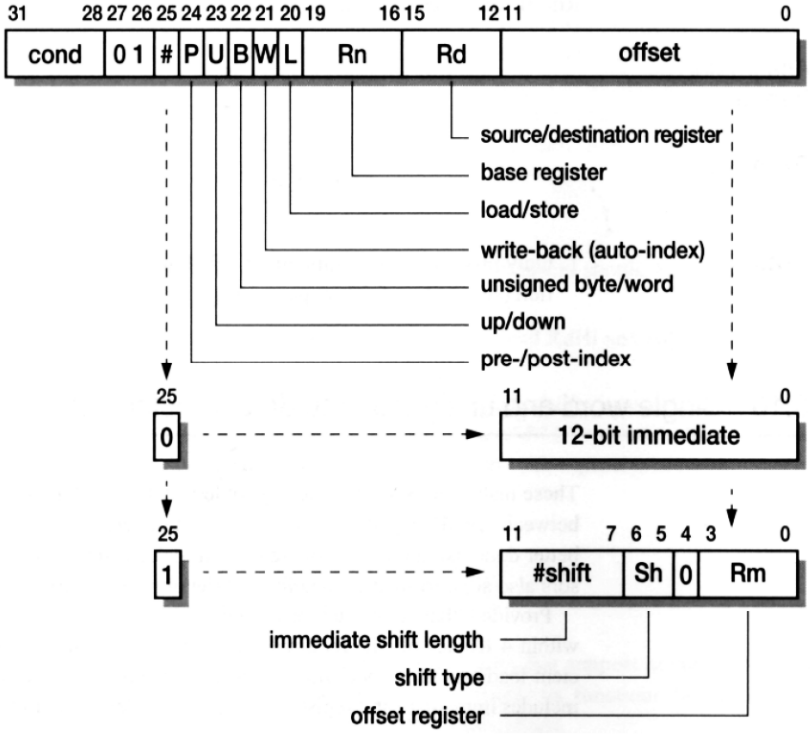
EL REGISTRO DE INSTRUCCIÓN (IR) – ARM7

- Ejemplo de instrucción genérica de Procesado de Datos:



El Registro de Instrucción (IR) – ARM7

- Ejemplo de instrucción genérica de Transferencia de Datos:



El Contador de Programa

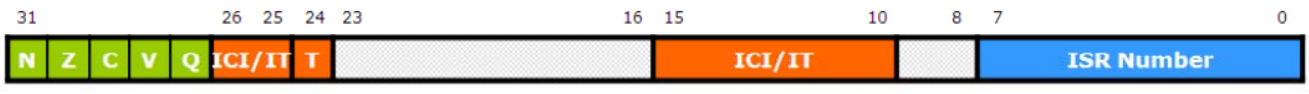
- Contiene en cada instante la dirección de la **siguiente instrucción** a capturar
- Se trata de un registro con las siguientes características:
  - Mismo tamaño que el bus de direcciones
  - Capacidad de ser modificado completamente por determinadas instrucciones
  - Capacidades especiales de "autoincremento" 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



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

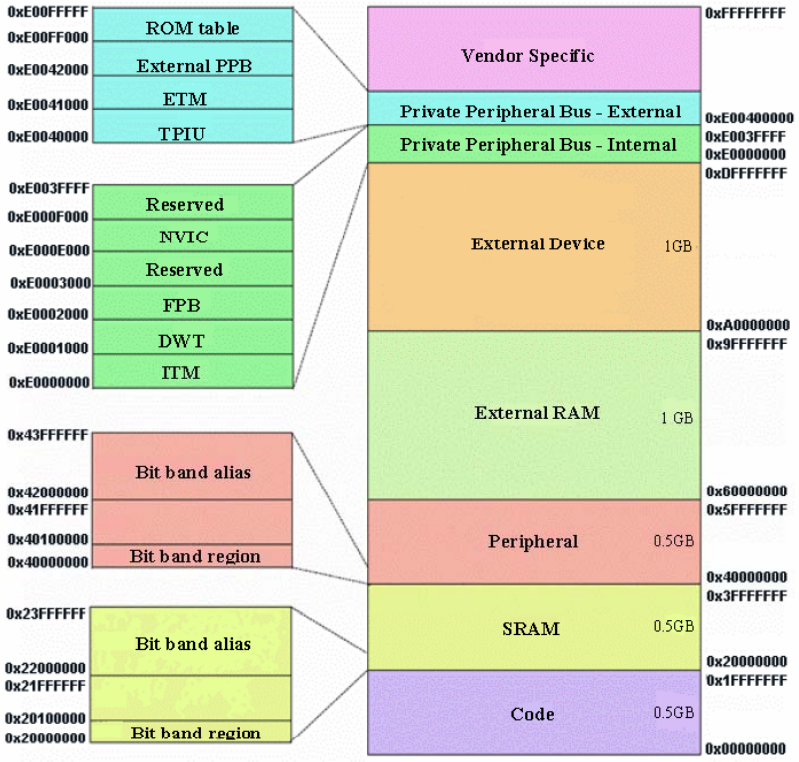
- Comunicación por buses (buses externos)
  - **Bus de Datos:** el que lleva el dato al que se está accediendo
    - Tiene el mismo número de líneas que el tamaño de la palabra de la CPU (aunque puede haber excepciones)
  - **Bus de Direcciones:** el que lleva la dirección del dato al que se quiere acceder
    - Tiene el mismo tamaño que el MAR y el PC
    - Cuanto más líneas se tenga, mayor será la capacidad de memoria que se le pueda conectar a la CPU
  - **Bus de Control:** el que contiene las distintas líneas que hacen posible un protocolo de comunicación con los chips de memoria
    - Por ejemplo, la señal de R/W, de AS#, o de CS#

- **Decodificador de Direcciones:**
  - Su misión es activar señales de selección de dispositivos (CS#) dependiendo del rango en el que se encuentre la dirección de memoria accedida
  - Para construirlo se pueden utilizar distintas estrategias:
    - Completa: cada CS# se obtiene utilizando todas las líneas de direccionamiento del micro. Hay una relación biunívoca entre dirección física accedida y contenido del bus de direcciones
    - Parcial: no todas las líneas de direcciones forman parte del proceso de decodificación. Existen direcciones físicas a las que se puede acceder con distintos contenidos del bus de direcciones
    - Por Bloque: se van definiendo bloques de direcciones y luego esos bloques se vuelven a decodificar (es una decodificación por pasos, o jerárquica).

- 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)



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) ⇐ MBR ; SP ⇐ SP-1
    - Una de lectura (PULL) sería
      - SP ⇐ SP+1 ; MBR ⇐ (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*

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.)