ARM

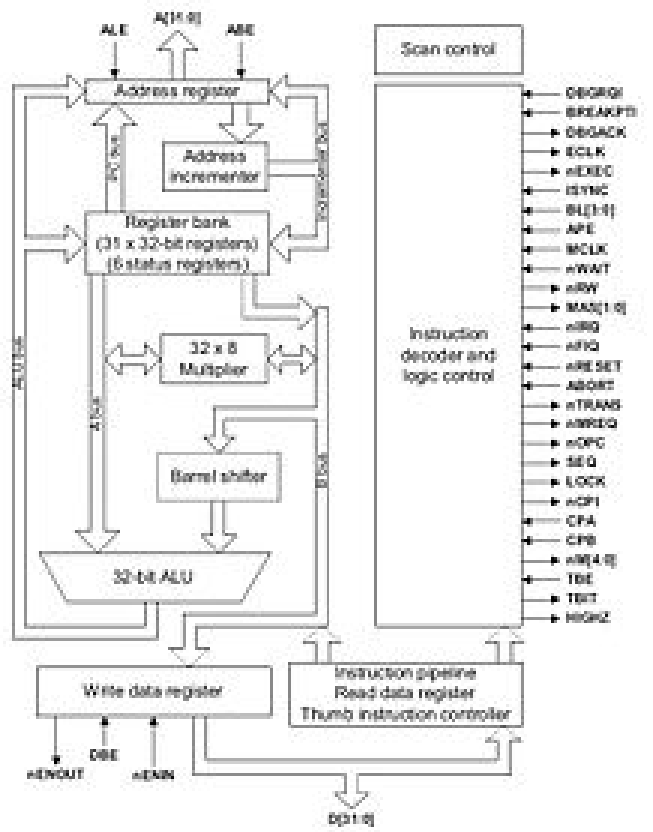
- El primer ARM fue desarrollado por Acorn RISC Machine entre 1983 y 1985
  - Creado a partir de un desarrollo de unos estudiantes de Berkeley
- En 1990, la empresa se convierte en Advanced RISC Machines Ltd
  - Compañía inglesa fundada por Apple Computer, Acorn Computer Group y VLSI Technology; actualmente ARM Limited
- Dedicada al desarrollo de procesadores RISC
- Cubre aproximadamente 75% del mercado mundial en procesadores RISC
- No hace procesadores, sólo los diseña
- Concede licencias de diseño a diversos fabricantes

- Microprocesador RISC
- 37 registros de 32 bits (17 ó 18 visibles)
- Memoria Caché (dependiendo de la aplicación)
- Bus tipo Von Newman (ARM7)
- Bus tipo Harvard (ARM9 y posteriores)
- Tipos de datos de 8/16/32 bits
- 6 ó 7 modos de operación
- Estructura simple
  - Alta velocidad
  - Bajo consumo de potencia
- Programas compatibles entre distintas familias ARM

- Instrucciones conceptualmente simples
- Transferencias memoria/registros, Load/Store
- Operaciones aritméticas entre registros
- Tamaño de instrucciones uniforme (32 bits)
- Pocos formatos para las instrucciones
- Pocos modos de direccionamiento

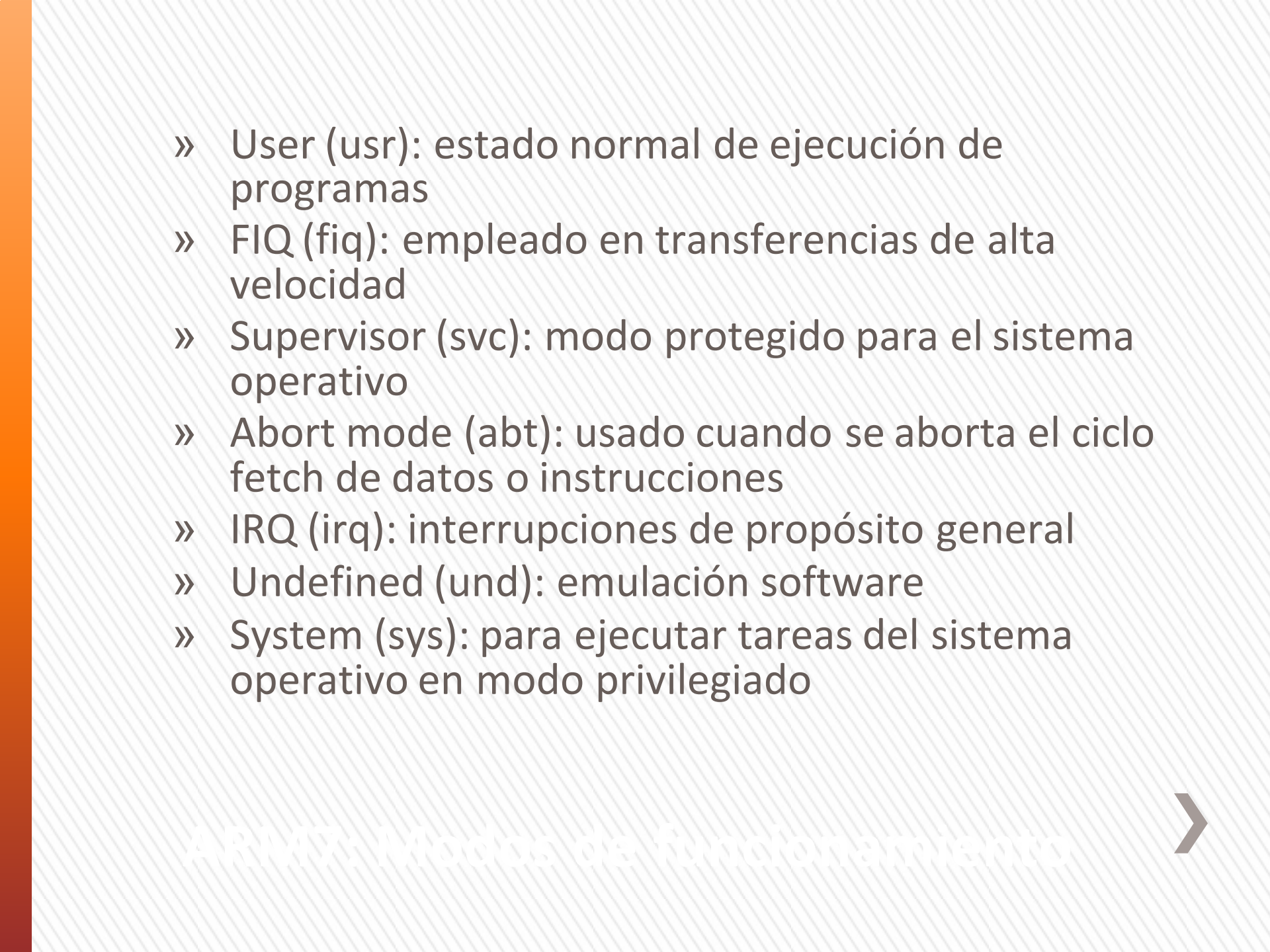
- Dual Master DMA controller
- DRAM controller
- Static Memory Controller
- Vectored Interrupt Controller
- UART
- GPIO
- Real-time Clock (RTC)
- Synchronous Serial Port (SSP)
- DC-DC controller
- Advanced Audio Codec Interface (AACI)
- LCD controller
- Media Card Interface

- Register Bank
  - 2 puertos de lectura + 1 de escritura de acceso a todos los registros
  - 1 puerto de lectura + 1 de escritura para acceder a R15, PC
- Barrel Shifter: desplaza o rota el contenido de los registros
- Address register + incrementer: almacena las direcciones de acceso a memoria
- Data Register: almacena datos en las transferencias con memoria
- Instruction decode & control

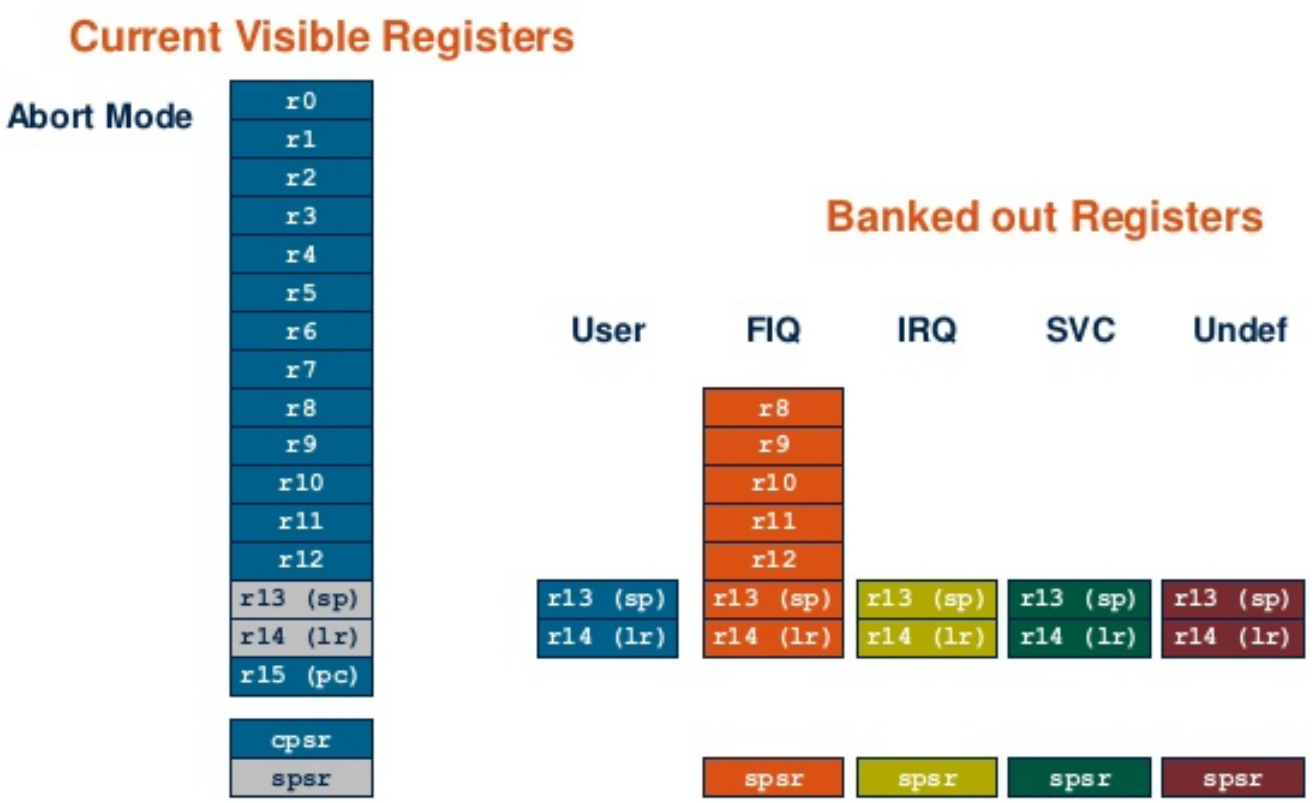


- Pipeline de 3 etapas:
  - Fetch: se obtiene la instrucción de memoria y se almacena en le pipeline de instrucciones
  - Decode: se decodifica la instrucción y se activan las señales de habilitación de ¨Datapath¨
  - Execute: operaciones y transferencias de datos ocupando el Datapath
- Todas las instrucciones ocupan el datapath durante al menos 1 ciclo
- El ciclo anterior activan las líneas de decodificación
- Durante el 1er ciclo de datapath se obtiene de memoria la siguiente instrucción
- Una instrucción de salto provoca el borrado del pipeline y un nuevo llenado
- En el 3er ciclo de una instrucción, se está cargando la segunda instrucción posterior

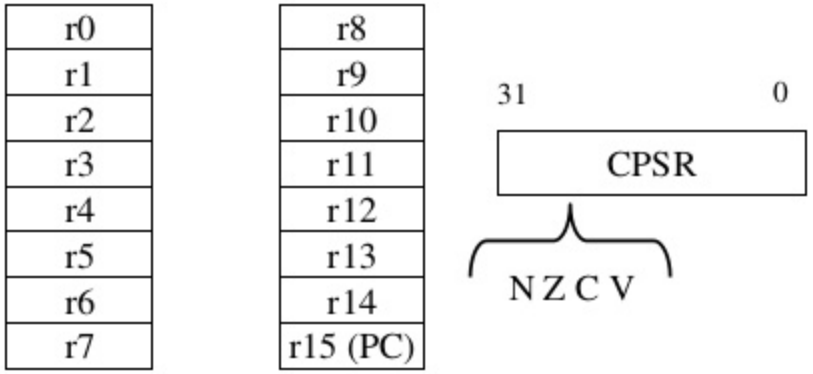
- User (usr): estado normal de ejecución de programas
- FIQ (fiq): empleado en transferencias de alta velocidad
- Supervisor (svc): modo protegido para el sistema operativo
- Abort mode (abt): usado cuando se aborta el ciclo fetch de datos o instrucciones
- IRQ (irq): interrupciones de propósito general
- Undefined (und): emulación software
- System (sys): para ejecutar tareas del sistema operativo en modo privilegiado



The ARM Register Set



ARM programming model

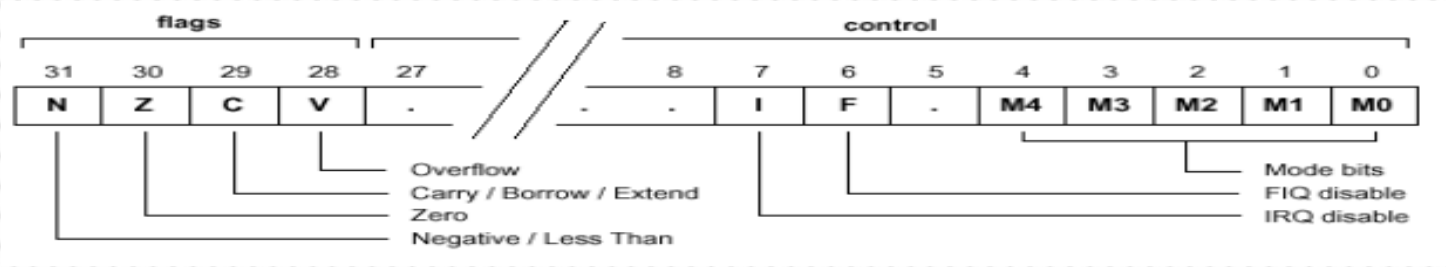


CPSR: Current Program Status Register
SPSR: Saved Program Status Register

- Registros de propósito general
  - Unbanked registers R0-R7: son comunes a todos los modos de funcionamiento
  - Banked registers R8-R14:
    - Se accede a distintos registros dependiendo del modo en que se encuentre
    - R8-R12, 2 banked physical registers
    - R13-R14, 6 banked physical registers
- R13 es el SP (stack pointer o puntero de pila)
  - No hay instrucciones específicas, excepto en modo Thumb
  - Se emplea para manejo de pila
  - Cada modo de funcionamiento tiene su propio SP que debe ser inicializado a una zona de memoria dedicada

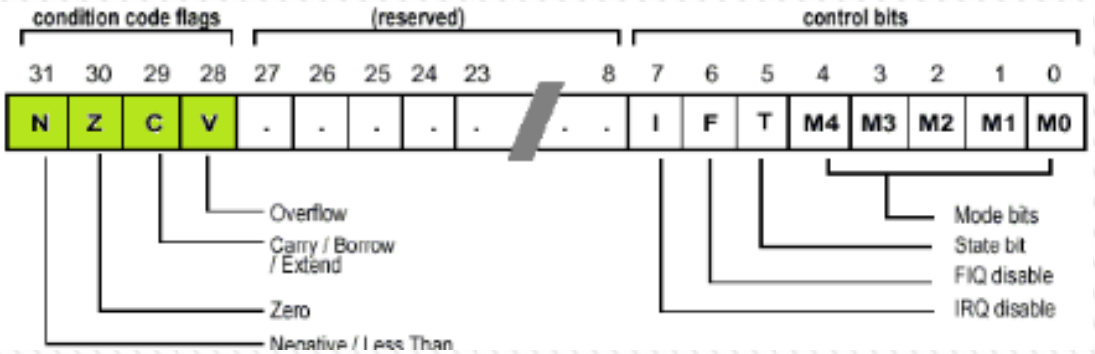
- R14 es el LR (Link Register)
  - Existe uno para cada modo de direccionamiento
  - Permite almacenar la dirección de retorno en una llamada a una rutina
  - Evita el almacenamiento del contador de programa en la pila
  - Proporciona un retorno rápido
  - Se emplea también para almacenar la dirección de retorno cuando se producen excepciones
- R15, es el PC (Program Counter)
  - Contiene la dirección de la instrucción que se va a ejecutar
  - Se puede emplear como cualquier otro registro de propósito general, salvo en determinadas excepciones
  - La lectura de PC mediante una instrucción, devuelve el valor de la dirección actual + 8 bytes
  - Debe tenerse en cuanta para realizar direccionamientos relativos
  - La escritura del PC provoca un salto a la dirección cargada
  - La dirección almacenada debe ser múltiplo de 4, ya que las instrucciones son de tamaño word (dirección par en modo Thumb)

- Registro de Estado CPSR (current program status register)
  - Accesible en todos los modos de funcionamiento
  - Contiene las banderas de condición
- SPSR (saved program status register)
  - Contiene una copia del CPSR cuando se entra en un modo privilegiado
  - Tienen este registro todos los modos excepto el user y system
- Formato del CPSR y SPSR

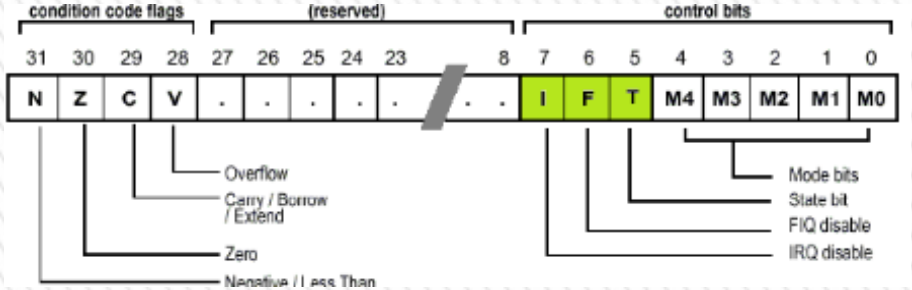


Banderas de condiciones

- **N**: bit de signo, 0 → positivo, 1 → negativo
- **Z**: bit de cero, 0 → resultado de operación ≠ 0, 1 → resultado de la operación = 0
- **C**: bit de acarreo
- **V**: bit de desbordamiento
- **Q**: desbordamiento/saturación (bit 27 en versiones E)



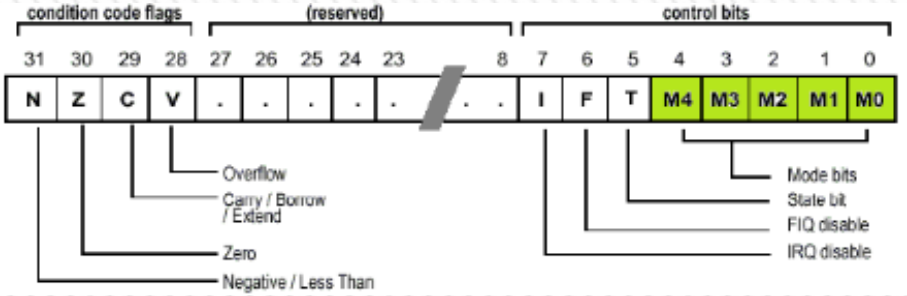
- Bits de control
  - Se modifican durante la ejecución de una excepción o por programa
- Bits de habilitación de interrupciones
  - I, puesto a 1, deshabilita las interrupciones IRQ
  - F, puesto a 1, deshabilita las interrupciones FIQ
- Bit T, selección del conjunto de instrucciones Thumb
  - T, 0 → ejecución ARM, 1 → ejecución Thumb
  - En versiones sin extensión Thumb, funciona como bit de traza



» Bits de modo

> M0, M1, M2, M3 y M4, representan el modo de operación del micro

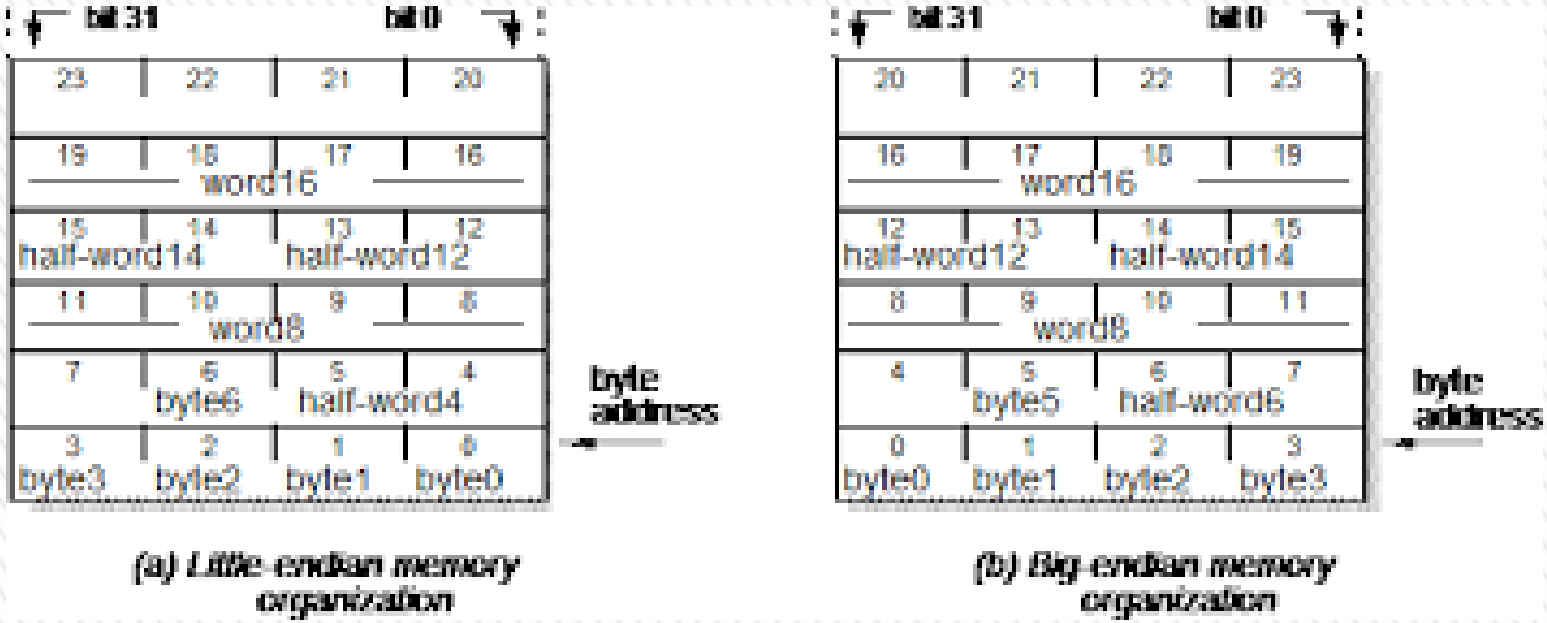
CPSR[4:0]	Modo	Uso	Registros
10000	User	Normal user code	User
10001	FIQ	Processing fast interrupts	_fiq
10010	IRQ	Processing standard interrupts	_irq
10011	SVC	Processing software interrupts (SWIs)	_svc
10111	Abort	Processing memory faults	_abt
11011	Undef	Handling undefined instruction traps	_und
11111	System	Running privileged operating systems tasks	user



- Byte, 8 bits
- Halfword, 16 bits
- Word, 32 bits
- Representaciones
  - Con signo → 0 .. $2^N - 1$, Con signo → C2, -2^{N-1} .. $2^{N-1} - 1$
- Almacenamiento en registros y operaciones en tamaño word
  - Posibilidad de extensión de signo
- Transferencias con memoria en todos los tamaños

Tamaños

» Organización de memoria ARM



- Organización de los datos en los registros
  - Las transferencias de datos de memoria a registros pueden ser tamaño byte, word o halfword
  - La transferencia de datos en tamaño inferior a word se almacenan en la parte de menor peso de los registros, rellenando con 0 la parte alta
  - Existen instrucciones para transferencia de bytes o halfwords con signo, en cuyo caso los bits de mayor peso serán una extensión del signo de dato transferido

- Permiten realizar operaciones aritméticas y lógicas sobre el contenido de los registros
- Típicamente requieren 2 operandos y generan un único resultado sencillo
- Características de estas instrucciones
  - Operandos fuente y destino de 32 bits (excepto ¨long multiply¨, 64 bits)
  - Los operandos son siempre registros o valores inmediatos
  - Se pueden especificar los operandos fuente y destino de forma independiente.
  - Ejemplo: Suma R1 + R2 y almacena resultado en R0
    - **ADD R0, R1, R2** ; R0 operando destino, R1 y R2 operandos fuente

Instrucciones aritméticas

- Suma binaria
- Resta binaria
- Resta binaria inversa
- Ejemplos de instrucciones
  - ADD R0, R1, R2 ; R0=R1+R2
  - ADC R0, R1, R2 ; R0=R1+R2+C → suma con acarreo
  - SUB R0, R1, R2 ; R0=R1-R2
  - SBC R0, R1, R2 ; R0= R2-R1+C-1 → resta con acarreo
  - RSB R0, R1, R2 ; R0=R2-R1 → resta binaria inversa
  - RSC R0, R1, R2 ; R0=R2-R1+C-1 → resta binaria inversa con acarreo

Instrucciones lógicas

- Producto lógico
- Suma lógica
- Suma exclusiva
- Borrado de un bit
- Ejemplos:
  - AND R0, R1, R2 ; R0=R1 and R2
  - ORR R0, R1, R2 ; R0=R1 or R2
  - EOR R0, R1, R2 ; R0=R1 xor R2
  - BIC R0, R1, R2 ; R0=R1 and not R2

- Instrucciones de movimiento de datos
  - Se diferencian de las de transferencia en que estas se realizan entre registros, y las otras con memoria. Se consideran de proceso de datos porque los modifican
  - Omiten el primer operando fuente
  - Copian el segundo operando de forma directa o en complemento a 1
  - Ejemplos:
    - MOV R0, R2 ; R0=R2
    - MVN R0, R2 ; R0=not R2
- Instrucciones de comparación
  - No generan resultado, sólo posicionan los códigos de condición (cc) del registro de estado
  - CMP (compare), CMN (compare negated), TST (test), TEQ (test equal)
  - Ejemplos:
    - CMP R1, R2 ; posiciona los cc en función del resultado de operación (R1-R2)
    - CMN R1, R2 ; posiciona los cc en función del resultado de operación (R1+R2)
    - TST R1, R2 ; posiciona los cc en función del resultado de operación (R1&R2)
    - TEQ R1, R2 ; posiciona los cc en función del resultado de operación (R1^R2)

» Operandos inmediatos

- Las instrucciones anteriores, además de con registros pueden operar con valores inmediatos
- Ejemplos:

```
ADD R0, R1, #5      ; R0=R1+5, el valor inmediato va precedido de #
ADD R3, R3, #1      ; R3 ++
ADD R8, R7, #&FF    ; R8=R7 & 0xFF, en ensamblador & significa hexadecimal, equivale a ¨0x¨ en C
```

- Los valores inmediatos se codifican en la misma instrucción, por lo que están limitados em tamaño
  - Valor inmediato = $(0 \ldots 255) * 2^{2n}$, $0 \leq n \leq 12$
- Cubre todas las potencias de 2, pero hay valores que no se pueden codificar

Instrucciones de desplazamiento y rotación

- No existen como tales, pero se pueden emplear para modificar un operando antes de realizar una de las operaciones anteriores

  ADD R1, R2, R3, LSL #5 ; R1=R2+R3*32

  R3 se desplaza 5 veces a la izquierda, lo que equivale a añadirle 5 ceros a la derecha, que es equivalente a multiplicar por $2^5 = 32$

- A pesar de lo complejo de la instrucción, el tiempo de ejecución es el mismo que las instrucciones simples (1 ciclo de reloj)
- Es posible indicar el número de desplazamientos mediante un registro

  ADD R1, R2, R3, LSL R4 ; $R1 = R1 + R2 + R3 * 2^{R4}$

- Operaciones de desplazamiento y rotación disponibles:
  - LSL, desplazamiento lógico a la izquierda, añade 0´s por la derecha
  - LSR, desplazamiento lógico a la derecha, añade 0´s por la izquierda
  - ASL, desplazamiento aritmético a la izquierda, añade 0´s por la derecha, es igual que LSL
  - ASR, desplazamiento aritmético a la derecha, repite el signo en los bits de la izquierda
  - ROR, rotación a derechas. El bit que se pierde se realimenta
  - RRX, rotación a derechas extendida a la bandera de acarreo

- Códigos de condición
  - Las instrucciones de comparación posicionan siempre los códigos de condición
  - El resto de instrucciones pueden posicionarlos o no a elección del programador
  - Esto se consigue añadiendo el sufijo ¨s¨ al nemónico correspondiente, ejemplo: ADDS R2, R2, R0 ; suma y posiciona las banderas de cc
  - Las operaciones de comparación y aritméticas posicionan todas las banderas
  - Las operaciones lógicas sólo posicionan las banderas N y Z, el resto permanecen intactos, excepto que se realice una rotación extendida, en cuyo caso modifican la bandera C
- Instrucciones de multiplicación
  - No soporta segundo operando un inmediato
  - El registro resultado no puede coincidir con el primer operando, ejemplo: MUL R1, R2, R3 ; R1=R2*R3
  - El resultado de la operación es en 64, pero el almacenamiento se realiza en un registro de 32. El valor almacenado consiste en los 32 bits bajos del resultado
  - Existe una variante de multiplicación y acumulación, ejemplo: MLA R1, R2, R3, R4 ; R1=R2*R3+R4

- Si se activa la opción ¨s¨ la bandera V no se afecta, y el C se modifica pero no tiene significado
- Para multiplicar por una constante se puede cargar la constante en un registro y a continuación multiplicar por el registro. Ejemplos de cómo multiplicar R0*27 y almacenar el resultado en R0
  - MOV R1, #27
  - MOV R2, R0
  - MUL R0, R1, R2 ; R0=R0*27
- En estos casos es más eficiente el empleo de instrucciones de suma con desplazamiento:
  - ADD R0, R0, R0, LSL #1 ; R0=R0*3
  - ADD R0, R0, R0, LSL #3 ; R0=R0*9

Instrucciones de transferencia

- Mueven los datos entre registros y memoria
- Load (memoria a registro) / Store (registro a memoria)
- Tipos básicos
  - Transferencia desde o hacia un único registro
    - Admite tamaños de operandos Byte, Word o Halfword
  - Transferencia desde o hacia múltiples registros
    - Se emplean para salvar o recuperar el contenido de varios registros en memoria
  - Instrucciones de intercambio, SWAP
    - Permite el intercambio de los datos entre un registro y una posición de memoria
    - Realiza una operación simultánea de load y store
- Modos de direccionamiento
  - Las formas de acceder a memoria son variadas, y están definidas por los diferentes modos de direccionamiento
  - La transferencia de los daos de memoria implican un registro, por lo que uno de los modos de direccionamiento empleados será directo a registro
  - Los accesos a memoria están basados todos ellos en direccionamiento indirecto por registro, en sus diferentes variantes

- Direccionamiento inmediato: es cuando se ingresa un dato, código ASCII, etc. De forma inmediata a un registro
- Direccionamiento directo: es cuando se realizan operaciones entre registros y localidades de memoria. Cabe resaltar que la localidad de memoria debe colocarse dentro de paréntesis para no confundirlo con un número.
- Direccionamiento por registro: operaciones entre registros, de cualquier tipo.

- Direccionamiento indirecto por registro: la dirección efectiva es la contenida en el registro empleado
  - LDR R0, [R1] ; carga en R0 el contenido de la posición apuntada por R1
  - STR R0, [R1] ; almacena el contenido de R0 en la posición apuntada por R1
- Inicialización del registro índice
  - Para asignar el valor inicial del registro índice se puede emplear una instrucción de movimiento de datos
  - Otra alternativa es emplear una pseudo instrucción del ARM, que carga un valor en el registro empleando como base el valor del PC

    ADR R1, Tabla 1 ; hace que R1 apunte a Tabla 1. ADR se traduce por una instrucción de suma o resta
- Ejemplo de transferencia sencilla

  ADR R1, Tabla1 ; R1 apunta a Tabla1

  ADR R2, Tabla2 ; R2 apunta a Tabla2

  LDR R0, [R1] ; Carga en R0 el contenido de la posición de Tabla 1

  STR R0, [R2] ; almacena el contenido de R0 en la posición Tabla2
- Para recorrer la tabla se pueden incrementar los registros mediante una instrucción de suma

  ADD R1, R1, #4 ; R1 apunta al siguiente word de la Tabla1, cada word ocupa 4 posiciones

  ADD R2, R2, #4 ; R2 apunta al siguiente word de la Tabla 2

- Direccionamiento indirecto por registro con desplazamiento
  - La dirección efectiva es la contenida por el registro empleado sumado con el valor inmediato adjunto, ejemplo: LDR R0, [R1, #4] ; carga en R0 el contenido de la posición apuntada por R1+4
  - En el ARM este modo es conocido como ¨direccionamiento preindexado¨
  - Al igual que en el ejemplo anterior, puede ser necesario modificar el valor de R1, esto puede realizarse empleando un modo de direccionamiento preindexado con autoindexado, ejemplo: LDR R0,[R1,#4]! ; Carga en R0 el contenido de la posición apuntada por R1+4 e incrementa el contenido en R1 en 4
  - El incremento de R1 no requiere un tiempo de ejecución adicional
- Direccionamiento postindexado: realiza el incremento en el registro, pero no emplea el desplazamiento en el direccionamiento
  - LDR R0, [R1], #4; Carga en R0 el contenido de la posición apuntada por R1 e incrementa el contenido de R1 en 4
- Selección del tamaño y modo de transferencia: las transferencias con memoria pueden realizarse en 3 tamaños y con o no sin signo. Esto se consigue añadiendo modificadores a las instrucciones ya vistas

```
LDRB R0, [R1]    ; Transferencia en tamaño Byte sin signo
LDRH R0, [R1]    ; Transferencia en tamaño Half-Word sin signo
LDRSB R0, [R1]   ; Transferencia en tamaño Byte con signo
LDRSH R0, [R1]   ; Transferencia en tamaño Half-Word con signo
```

» Transferencia hacia múltiples registros

- > Carga el contenido de posiciones consecutivas de memoria en varios registros, ejemplo: LDMIA R0, {R1, R2, R5} ; carga en R1 el contenido de la posición apuntada por R0, en R2 el apuntado por R0+4 y en R5 el apuntado por R0+8
- > El orden en que se especifiquen los registros es irrelevante, se carga el dato de la posición más baja de memoria en el registro de menor orden y el resto en los registros de orden inmediato superior
- > Las transferencias se realizan en 32 bits, por lo que R0 debe apuntar a una posición múltiplo de 4

» Resumen de instrucciones de transferencia múltiples

		Ascending		Descending	
		Full	Empty	Full	Empty
Increment	Before	STMIB, STMFA			LDMIB, LDMED
	After		STMIA, STMEA	LDMIA, LDMFD	
Decrement	Before		LDMDB, LDMEA	STMDB, STMFD	
	After	LDMDA, LDMFA			STMDA, STMED

Control de la pila

- No existen instrucciones específicas para el control de la pila, por lo que se emplean las de transferencia de datos
- Una pila tiene un funcionamiento LIFO (last in first out) por lo que hay que almacenar y recuperar datos en orden inverso
- Las pilas funcionan de diferente modo dependiendo del microprocesador, dado que aquí el control es total, se puede generar una pila que almacene los datos en sentido ascendente o descendente, es decir, que incremente la posición de memoria cada vez que almacene datos o la decremente
- Del mismo modo, el puntero de pila apunta al último dato almacenado o al primer espacio libre dentro de la pila (full stack y empty stack, respectivamente)
- Acceso a la pila
  - Por convenio se emplea el registro R13 como Stack Pointer
  - Almacenamiento de un dato en la pila, ejemplo STR R0, [R13, #4] ; Almacena el contenido de R0 en la dirección R13+4 e incrementa R13 en 4
  - Recuperación de un dato de la pila, ejemplo: LDR R0, [R13], #-4 ; Carga en R0 el contenido de la dirección R3 y decrementa R13 en 4
  - En este ejemplo el puntero contiene la dirección del último dato escrito (full stack), se trata por tanto de una pila ¨full ascending¨
  - Combinando las instrucciones se pueden conseguir los otros 3 modos posibles: full descending, empty ascending y empty descending

Ejemplo de acceso a la pila

- Se puede realizar el acceso con instrucciones de acceso múltiple
  - STMFD R0!, {R2-R9} ; almacena los registro R2 a R9 en la pila (full descending)
  - LDMFD R0!, {R2-R9} ; recupera los datos de la pila
- Tenga en cuenta que los modos de almacenamiento y recuperación deben ser los duales para que la pila quede en el estado inicial después de recupera los datos
- Esta instrucción, evidentemente, no se ejecuta en un ciclo de reloj ni aun en el caso de arquitecturas Harvard

Instrumentos de control

- Este conjunto de instrucciones permiten determinar qué instrucción se ejecutará, pero no realizan modificación ni transferencia de datos
- Dentro de este grupo de instrucciones están las instrucciones de salto
- Tipos de salto
  - Salto incondicional: se realiza siempre que aparece la instrucción, ejemplo: B Etiqueta ; salta a la instrucción señalada por ¨etiqueta¨
  - Salto condicional: se ejecuta si se cumple una condición basada en las bandera del registro de estado. Ejemplo: BEQ Etiqueta ; salta a la instrucción señalada por ¨etiqueta¨ si está activo la bandera de cero

» Condiciones de salto

Branch	Interpretación	Uso normal
B BAL	Unconditional Always	Always take this branch Always take this branch
BEQ	Equal	Comparison equal or zero result
BNE	Not equal	Comparison not equal or non-zero result
BPL	Plus	Result positive or zero
BMI	Minus	Result minus or negative
BCC BLO	Carry clear Lower	Arithmetic operation did not give carry-out Unsigned comparison gave lower
BCS BHS	Carry set Higher or same	Arithmetic operation gave carry-out Unsigned comparison gave higher or same
BVC	Overflow clear	Signed interger operation; no overflow occurred
BVS	Overflow set	Signed interger operation; overflow occurred
BGT	Greater than	Signed integer comparison gave greater than
BGE	Greater or equal	Signed interger comparison gave greater or equal
BLT	Less than	Signed integer comparison gave less than
BLE	Less or equal	Signed integer comparison gave less than or equal
BHI	Higher	Unsigned comparison gave higher
BLS	Lower or same	Unsigned comparison gave lower or same

» Ejemplo 1: ejecución de un conjunto de instrucciones varias veces, equivalente a un bucle ¨for (i=0; i<10; i++) {…..}¨

```
        MOV R0, #0          ; Inicialización del contador, i=0
Bucle ….                    ; conjunto de instrucciones que se repiten en el
        ….                  ; bucle
        ADD R0, R0, #1      ; Incremento del contador, i++
        CMP R0, #10         ; comparación del contador, i<10
        BNE  Bucle          ; salto al principio del bucle mientras se cumpla la condición
```

» Ejemplo 2: ejecución condicional de una instrucción ¨if (R0=5) {R1=2*R1-R2}¨

```
        CMP R0, #5
        BNE  Siguiente      ; if
        ADD R1, R1, R1      ; instrucciones que se ejecutan
        SUB  R1, R1, R2     ; si se cumple la condición
Siguiente …..
```

» Otra variante:

```
        CMP  R0, #5
        BNE   Siguiente             ; if
        RSB   R1, R2, R1, LSL #1    ; instrucciones
Siguiente …..
```

- Las instrucciones de salto vacían el pipeline, y ralentizan la ejecución del programa
- Existe otra posibilidad de ejecución condicional, ejemplo: para una instrucción ¨if (R0=5){R1=2*R1-R2}¨

```
CMP     R0, #5
ADDNE   R1, R1, R1 ; instrucciones que se ejecutan
SUBNE   R1, R1, R2 ; si se cumple la condición
…..
```

- La estructura del ARM permite ejecutar cada instrucción de forma condicional, reflejando la condición dentro del código de operación
- Ejemplo de ejecución condicional con varias condiciones: ¨if ((a=b) & (c=d)) e++¨

```
CMP     R0, R1        ; if (a=b)
CMPEQ R2, R3          ; {If (c=d)
ADDEQ  R4, R4, #1  ; e++}
….
```

Llamadas a subrutinas

- Cuando se realiza la llamada a una rutina hay que almacenar la dirección de retorno, que coincide con la dirección de la instrucción siguiente a la de llamada
- Muchos microprocesadores almacenan la dirección de retorno en la pila
- El ARM no tiene control automático de pila, por lo que no almacenará la instrucción de retorno en ella
- La llamada a una rutina se puede realizar con la instrucción de salto B
- Otro modo de llamar a una subrutina es emplear BL
  - BL, Branch and Link, hace una copia de la dirección de la siguiente instrucción en el registro de enlace R14
- El ARM no tiene instrucciones de retorno de subrutinas, el retorno se realiza copiando el contenido del R14 en el PC
- Ejemplo de llamada a subrutina

```
    BL   Subrutina    ; salta a la subrutina y copia la dirección de retorno en LR (R14)
    …..               ; dirección de retorno
Subrutina ….
    …..
    MOV  PC, R14      ; copia el contenido de Link register en PC
```

- Anidamiento de subrutinas
  - Dado que sólo hay un registro R14 en cada modo, si se requiere anidar rutinas, es necesario almacenar su contenido en la pila para recuperarlo posteriormente
  - Ejemplo:

```
            BL   Subrutina_1   ; llamada a la subrutina 1
            …                  ;
Subrutina_1  STMFD  R13!, {R0-R2, R14} ; almacenamiento en pila de los registros que se usan en la subrutina_1 y de el LR
            ….
            BL    Subrutina_2  ; llamada a la subrutina 2
```

- Otra opción de retorno es recuperar el contenido del PC directamente desde la pila. Ejemplo:

```
Subrutina_1  STMFD   R13!, {R0-R2, R14}    ; almacenamiento en pila de los regis-
            …                              ; tros que se usan en la Subrutina_1 y del LR
            BL        Subrutina_2
            ….
            LDMFD  R13!, {R0-R2, PC}       ; recuperación de los registros R0, R1 y R2,
                                           ; y retorno de subrutina
```

  - Nótese que en la recuperación de los registros se ha sustituido R14 por R15(PC), por lo que se provoca el retorno de forma automática desde la pila, omitiendo el paso de recuperar el contenido de R14

- Ejecución de rutinas en modo supervisor
  - Para ejecutar una rutina en modo supervisor, se emplean las interrupciones de software ¨swi¨, también conocidas como llamadas de supervisor
  - Generalmente, realizan llamadas a rutinas del sistema operativo

Materiales explosivos

- Se emplean para tratar eventos que se producen durante la ejecución de un programa
- En algunos microprocesadores, incluido el ARM, el concepto de excepción incluye las interrupciones de hardware y software
- Las interrupciones de software no se pueden considerar como un evento inesperado, pero aún así se consideran excepciones por el tratamiento que se realiza de ellas
- El reset del microprocesador también se suele considerar como una excepción por coincidir su tratamiento con el de las excepciones en general
- Se dividen en 3 grupos:
  - Excepciones generadas directamente por la ejecución de una instrucción
    - Interrupciones de software, SWI
    - Instrucciones no definidas
    - Prefetch aborts, debido a un error en el acceso a memoria al buscar un código de operación
  - Excepciones generadas como efectos secundarios en la ejecución de una instrucción
    - Data aborts, debido a un error en el acceso a memoria al buscar o almacenar un dato
  - Excepciones generadas externamente
    - Reset, IRQ, FIQ

- Proceso de tratamiento de una excepción
  - Todas las excepciones tienen el mismo tratamiento excepto la de reset
  - Cuando llega una excepción
    - Se finaliza la ejecución de la instrucción en curso
    - La siguiente instrucción a ejecutar será la correspondiente a la rutina de la excepción asociada
    - Se cambia el modo de operación al modo correspondiente a la excepción que se produce (SVC, undefined, abort, FIQ, IRQ)
    - Se salva el contador de programa en el R14, del modo al que se ha cambiado
    - Se salva el CPSR en el SPSR correspondiente
    - Se desactivan las interrupciones IRQ siempre, y también las FIQ en caso de que sea la fuente de excepción
    - Se carga en el PC el vector correspondiente al modo en el que se entra
- Reset: al activar la terminal reset, se tiene:
  - Los registros R14 y SPRS toman valores impredecibles
  - Se activa el modo supervisor
  - Se activa el modo ARM
  - Se inhabilitan las interrupciones IRQ y FIQ
  - Se carga en el PC el vector de reset

» Tabla de vectores de excepción

Exception	Mode	Vector address Normal	High
Reset	SVC	0x00000000	0xFFFF0000
Undefined instruction	UND	0x00000004	0xFFFF0004
Software interrupt (SWI)	SVC	0x00000008	0xFFFF0008
Prefetch abort (instruction fetch memory fault)	Abort	0x0000000C	0xFFFF000C
Data abort (data access memory fault)	Abort	0x00000010	0xFFFF0010
IRQ (normal interrupt)	IRQ	0x00000018	0xFFFF0018
FIQ (fast interrupt)	FIQ	0x0000001C	0xFFFF001C

» En la dirección de cada vector se pone una instrucción de salto a la dirección de la rutina de tratamiento de la excepción, excepto en las interrupciones FIQ, que al ser el último vector, puede empezar el código de forma inmediata, como en el ahorro de tiempo

- Dentro de la rutina de tratamiento, se suelen almacenar en la pila particular los registros que se vayan a emplear, a excepción de las FIQ, que al tener duplicados los registros R8 a R12, puede hacer uso de ellos sin salvarlos previamente
  - Los punteros de pila se deben inicializar, en modo supervisor en el arranque
- En el retorno de las excepciones se debe:
  - Recuperar el contenido original de los registros empleados en la rutina
  - Restaurar el CPSR a partir del SPSR
  - Restaurar el valor del PC a su valor antes de la ejecución de la rutina
- Las 2 últimas operaciones deben llevarse a cabo simultáneamente, de otro modo no sería posible reestablecer su contenido original

- Existen instrucciones que permiten realizar las 2 operaciones simultáneamente, y que difieren según el modo en que se encuentre el micro
  - Retorno de una interrupción software o instrucción indefinida. Ejemplo: MOVS PC, R14 ; la S asociada al PC fuerza que se vuelque el contenido del SPSR al CPSR
  - Retorno de una interrupción (IRQ, FIQ), o una excepción prefetch aborts. Ejemplo: SUBS PC, R14, #4 ; retorna el programa en la instrucción siguiente.
  - Retorno de una excepción data abort. Ejemplo: SUBS PC, R14, #8 ; esto permite volver a ejecutar la instrucción que dio lugar al error
  - El primer tipo no requiere modificación del contenido de R14 por ser excepciones esperadas, en cuyo caso se almacena en R14 el valor de la dirección de retorno
  - Cada modo de funcionamiento está asociado a una excepción, menos el modo system
  - Para acceder al modo system debe modificarse el CPSR desde un modo supervisor, ya que desde el modo user el registro de estado no se puede modificar

» Prioridad de las excepciones

> Algunas excepciones existentes no pueden producirse simultáneamente, pero en el caso de que varias se produjeran en el mismo instante, existen prioridades preestablecidas:

Prioridad	Excepción
1. Máxima prioridad	Reset
2.	Data Abort
3.	FIQ
4.	IRQ
5.	Prefetch Abort
6. Mínima prioridad	Undefined instruction (SWI)