Introduction to ARM

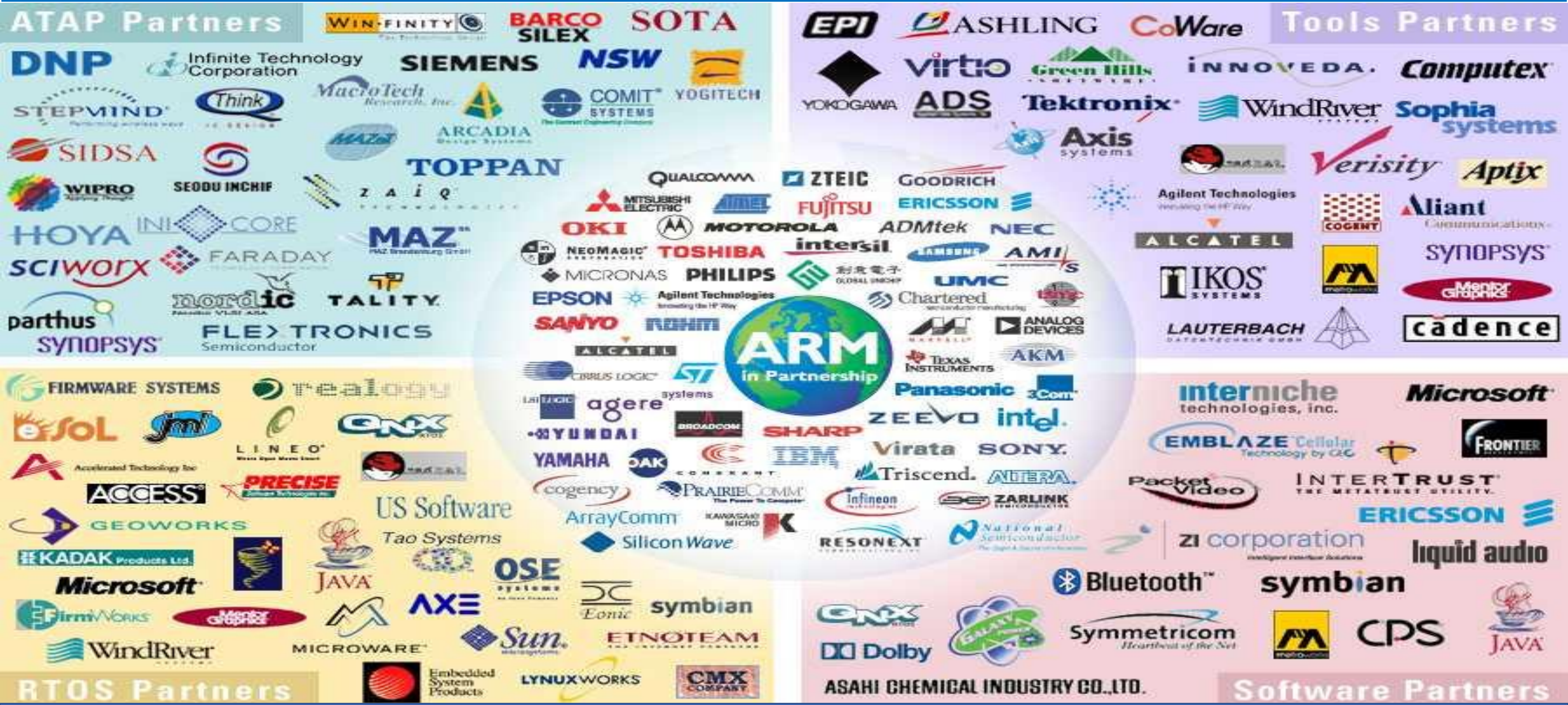
ARM

- Founded in November 1990
  - Spun out of Acorn Computers

- Designs the ARM range of RISC processor cores
- Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers.
  - ARM does not fabricate silicon itself

- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware, application software, bus architectures, peripherals etc

ARM Partnership Model



ARM Powered Products

CISC vs RISC

- RISC machine
  - *Pipelining* effectively provides single cycle operation for many instructions
  - Thumb-2 configuration employs both 16 and 32 bit instructions

CISC	*RISC*
Many instructions	Few instructions
Instructions have varying lengths	Instructions have fixed lengths
Instructions execute in varying times	Instructions execute in 1 or 2 bus cycles
Many instructions can access memory	Few instructions can access memory • Load from memory to a register • Store from register to memory
In one instruction, the processor can both • read memory and • write memory	No one instruction can both read and write memory in the same instruction
Fewer and more specialized registers. • some registers contain data, • others contain addresses	Many identical general purpose registers
Many different types of addressing modes	Limited number of addressing modes • register, • immediate, and • indexed.

ARM Features

- Based upon RISC Architecture with enhancements to meet requirements of Embedded applications.
- A large uniform file size.
- Load-Store Architecture, where data processing operations operate on register content only.
- Uniform and Fixed length instructions.
- Good Speed/Power consumption ratio.
- High Code Density.

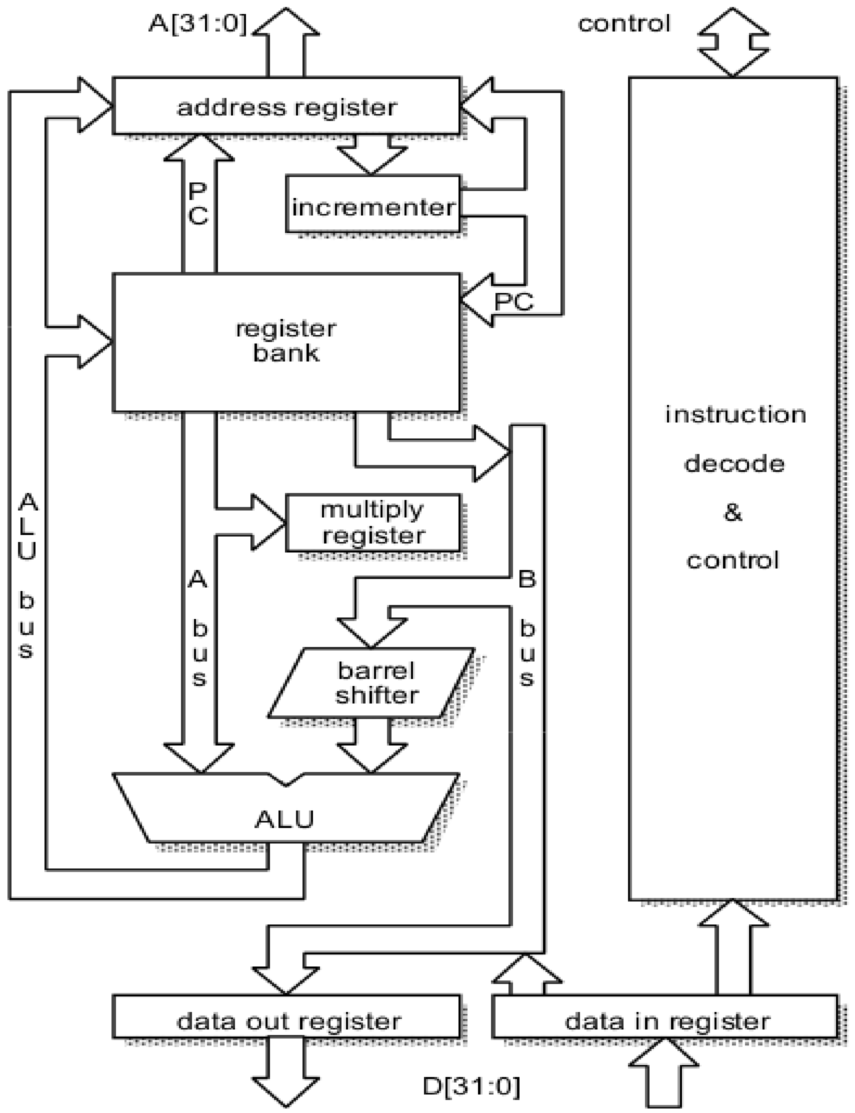
Intellectual Property

- **ARM provides hard and soft views to licencees**
  - RTL and synthesis flows
  - GDSII layout
- **Licensees have the right to use hard or soft views of the IP**
  - soft views include gate level netlists
  - hard views are DSMs
- **OEMs must use hard views**
  - to protect ARM IP

ARM Architecture Versions

Core	Architecture
ARM1	v1
ARM2	v2
ARM2as, ARM3	v2a
ARM6, ARM600, ARM610	v3
ARM7, ARM700, ARM710	v3
ARM7TDMI, ARM710T, ARM720T, ARM740T	v4T
StrongARM, ARM8, ARM810	v4
ARM9TDMI, ARM920T, ARM940T	V4T
ARM9E-S, ARM10TDMI, ARM1020E	v5TE
ARM10TDMI, ARM1020E	v5TE
ARM11 MPCore, ARM1136J(F)-S, ARM1176JZ(F)-S	v6
Cortex-A/R/M	v7

ARM Architecture



Core Data Path

- Data items are placed in register file
  - No Data processing instructions directly manipulate data in memory.
- Instructions typically use two source registers and single result or destination registers.
- A Barrel Shifter on the data path can pre-process data before it enters ALU.
- Increment/Decrement logic can update register content for sequential access independent of ALU.

ARM Architecture

ARM Programmers Model

Data Sizes and Instruction Sets

- The ARM is a 32-bit architecture.

- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)

- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set

ARM Processor Modes

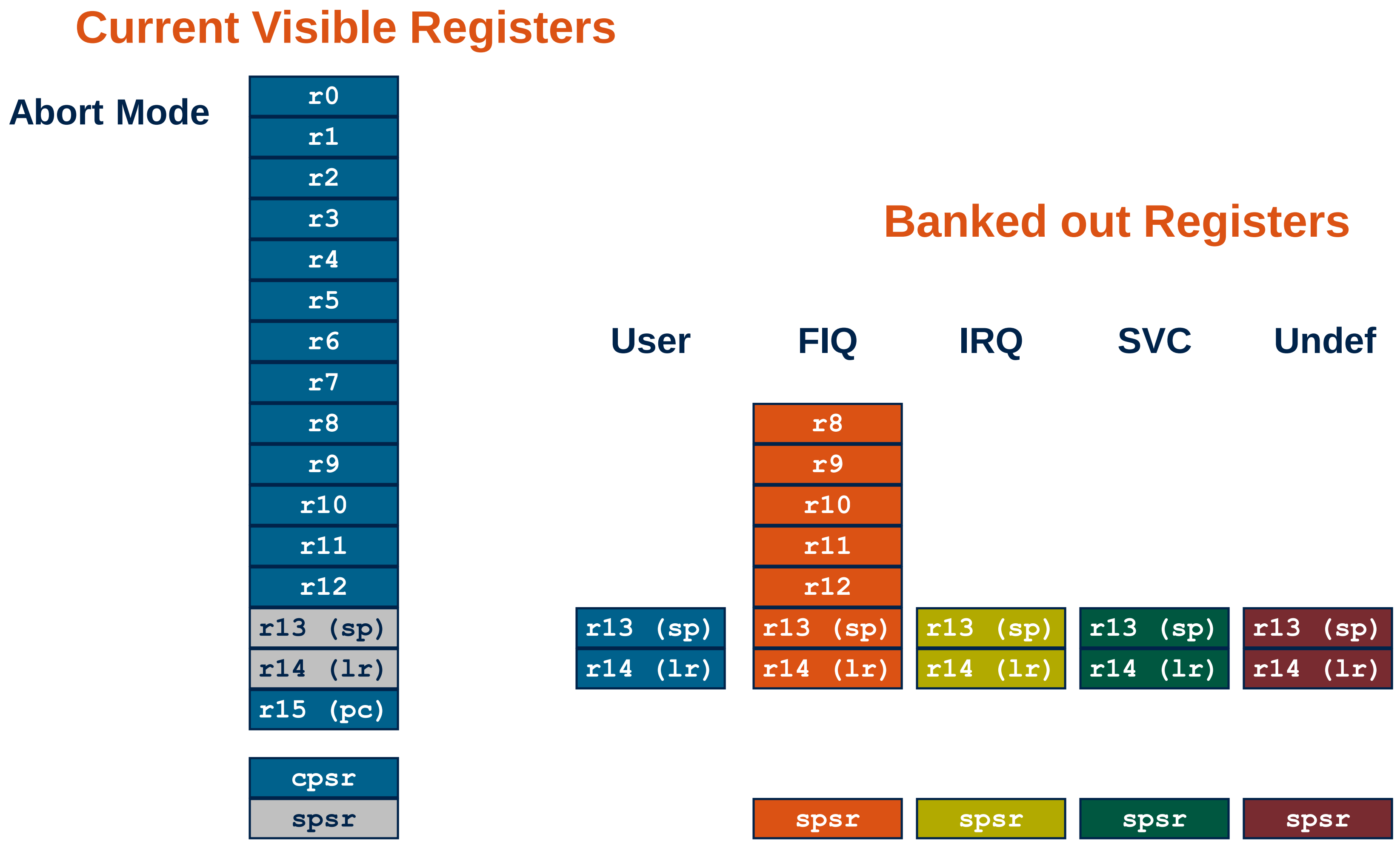
- Processor modes determines,
  - Which registers are active, and
  - Access rights to CPSR register itself.
- Each processor mode is either
  - Privileged: full read-write access to the CPSR. (abort, fast interrupt request, interrupt request, supervisor, system and undefined)
  - Non-Privileged: only read access to the control filed of CPSR but read-write access to the condition flags. (user)

ARM Processor Modes

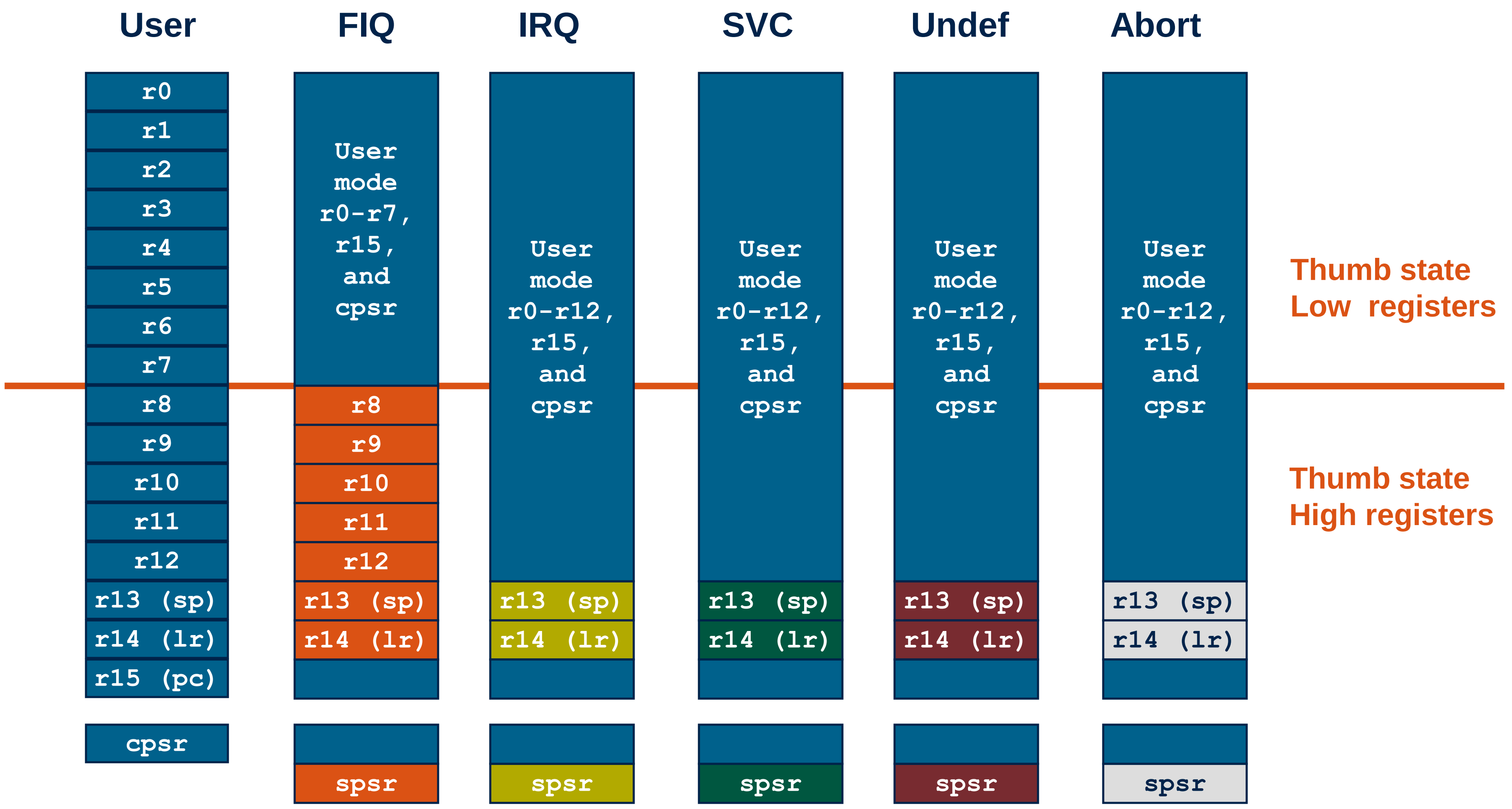
<u>The ARM has seven basic operating modes:</u>

- **user:** unprivileged mode under which most tasks run
- **FIQ:** entered when a high priority (fast) interrupt is raised
- **IRQ** entered when a low priority (normal) interrupt is raised
- **supervisor:** entered on reset and when a Software Interrupt instruction is executed. State after reset and generally the mode in which OS kernel executes,
- **abort:** used to handle memory access violations
- **undef:** used to handle undefined instructions
- **system:** privileged mode using the same registers as user mode.

The ARM Register Set



Register Organization Summary



User
FIQ
IRQ
SVC
Undef
Abort
r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
r15 (pc)
cpsr
User mode r0-r7, r15, and cpsr
r8
r9
r10
r11
r12
r13 (sp)
r14 (lr)
spsr
User mode r0-r12, r15, and cpsr
r13 (sp)
r14 (lr)
spsr
User mode r0-r12, r15, and cpsr
r13 (sp)
r14 (lr)
spsr
User mode r0-r12, r15, and cpsr
r13 (sp)
r14 (lr)
spsr
User mode r0-r12, r15, and cpsr
r13 (sp)
r14 (lr)
spsr
Thumb state Low registers
Thumb state High registers

Note: System mode uses the User mode register set

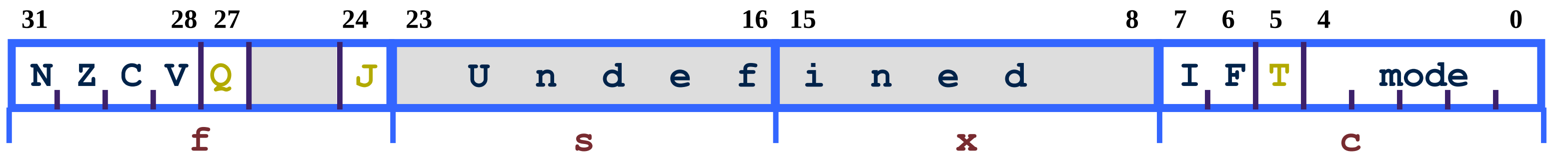
The Registers

- ARM has 37 registers all of which are 32-bits long.
  - 1 dedicated program counter
  - 1 dedicated current program status register
  - 5 dedicated saved program status registers
  - 30 general purpose registers

- The current processor mode governs which of several banks is accessible. Each mode can access
  - a particular set of r0-r12 registers
  - a particular r13 (the stack pointer, sp) and r14 (the link register, lr)
  - the program counter, r15 (pc)
  - the current program status register, cpsr

  Privileged modes (except System) can also access
  - a particular spsr (saved program status register)

Program Status Registers



- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Sticky Overflow flag - Q flag
  - Architecture 5TE/J only
  - Indicates if saturation has occurred
- J bit
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state

Interrupt Disable bits.

I = 1: Disables the IRQ.

F = 1: Disables the FIQ.

T Bit

Architecture xT only

T = 0: Processor in ARM state

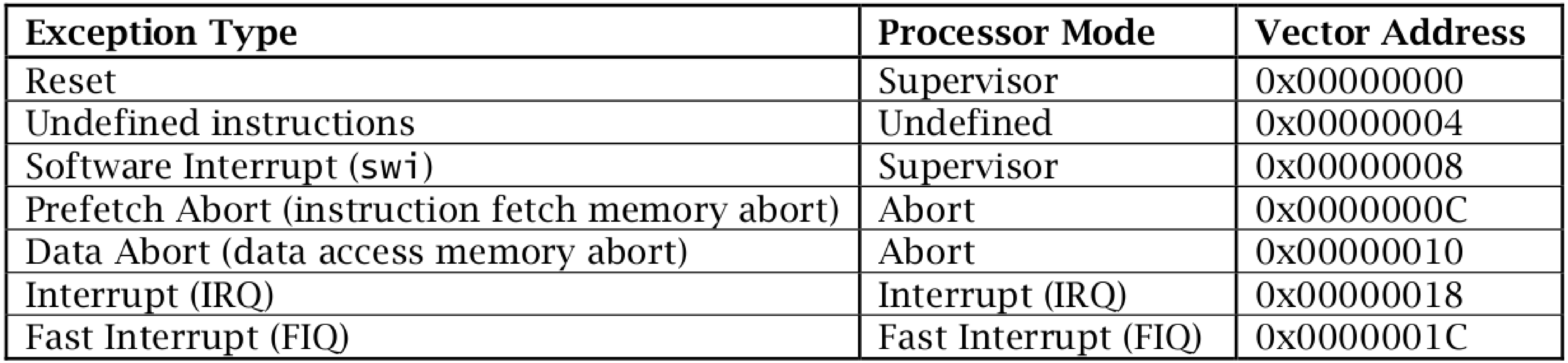
T = 1: Processor in Thumb state

Mode bits

Specify the processor mode

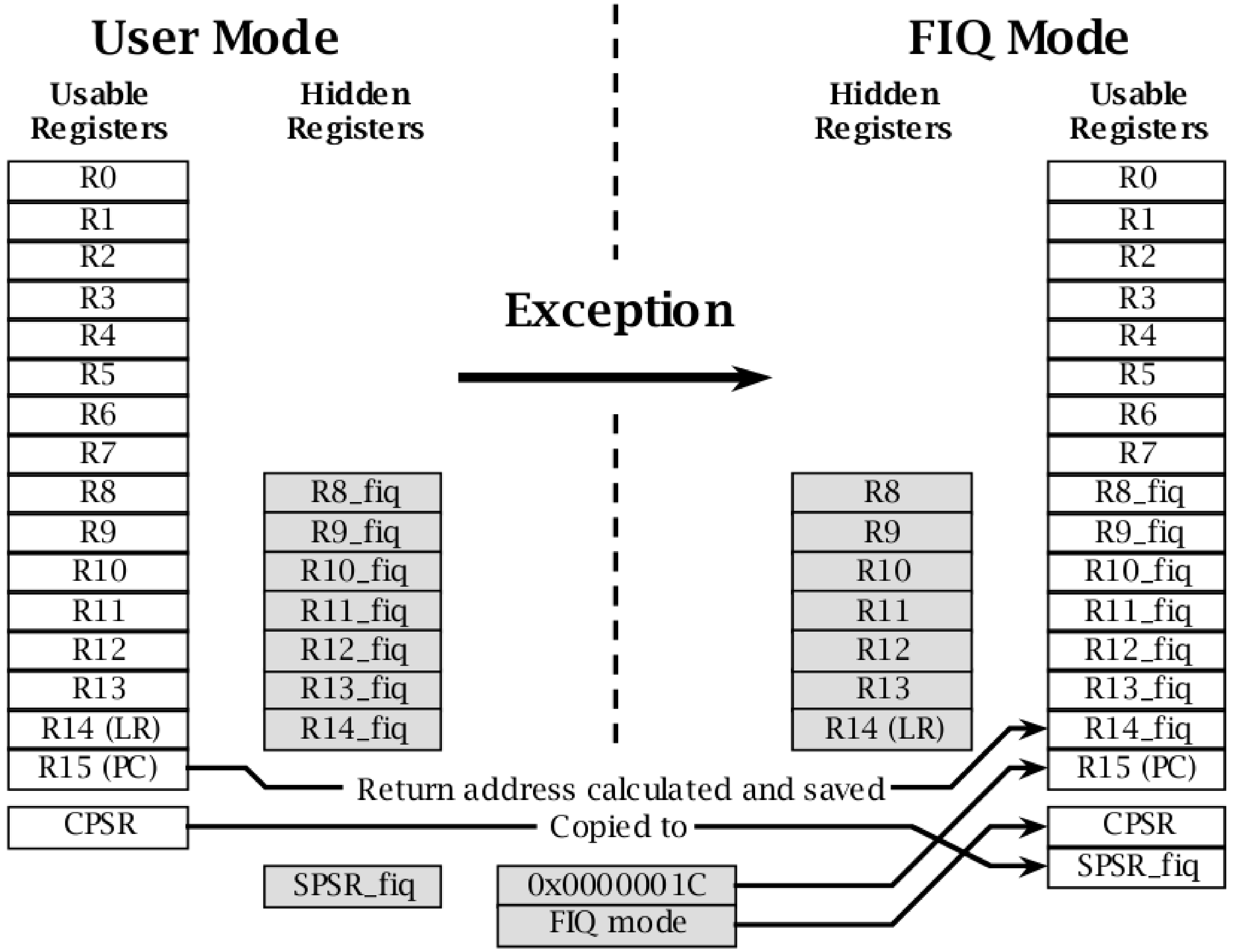
ARM Exceptions

- Interrupts generated by some peripheral device,
- An attempt to execute an undefined or unimplemented instruction,
- A software-generated interrupt, via the swi instruction.



Exception Type	Processor Mode	Vector Address
Reset	Supervisor	0x00000000
Undefined instructions	Undefined	0x00000004
Software Interrupt (`swi`)	Supervisor	0x00000008
Prefetch Abort (instruction fetch memory abort)	Abort	0x0000000C
Data Abort (data access memory abort)	Abort	0x00000010
Interrupt (IRQ)	Interrupt (IRQ)	0x00000018
Fast Interrupt (FIQ)	Fast Interrupt (FIQ)	0x0000001C

Exception Handling



Switching from User Mode to FIQ Mode on a Fast Interrupt Exception

Exception Handling

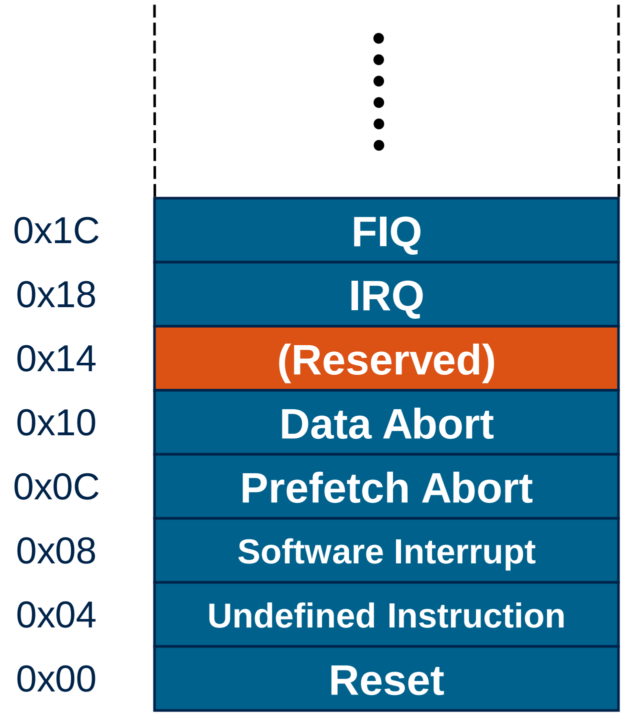
To handle an exception, the ARM processor:

1. copies the address of the next instruction (the return address), or the return address plus some offset, into the appropriate LR register,
2. copies the CPSR into the appropriate SPSR,
3. sets the CPSR mode bits to the processor mode corresponding to the exception,
4. enforces ARM state by setting bit 5 (the T bit) of CPSR to zero,
5. possibly disables fast interrupts by setting bit 6 of CPSR to one (only for FIQ exceptions),
6. disables normal interrupts by setting bit 7 (the I bit) of CPSR to one, and
7. loads the address of the exception vector into the Program Counter PC.

Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

This can only be done in ARM state.



Vector Table

Vector table can be at `0xFFFF0000` on ARM720T and on ARM9/10 family devices