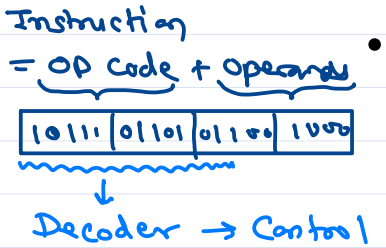
Advanced Micro-controllers - ARM

DESD @ Sunbeam Infotech

Agenda

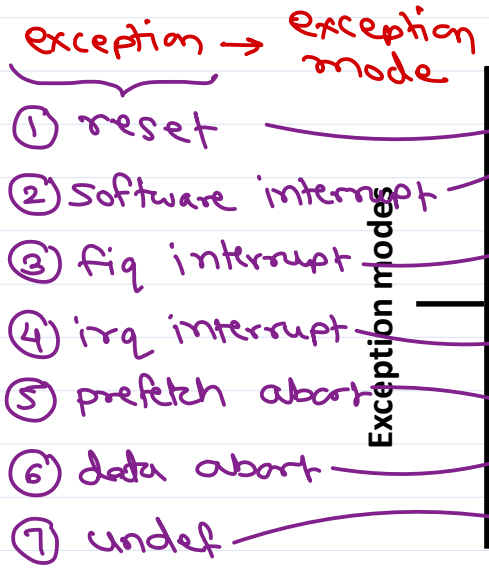
- Introduction
- ARM Architecture Overview
- ARMv7-AR Architecture
  - Programmer's Model
  - Memory Systems
- ARMv7-M Architecture
  - Programmer's Model
  - Memory Systems
  - Floating Point Extensions
- ARM System Design
- Software Development Tools

Processor Modes



- ARM has seven basic operating modes
  - Each mode has access to its own stack space and a different subset of registers
  - Some operations can only be carried out in a privileged mode

Software Interrupt is used for OS System calls.



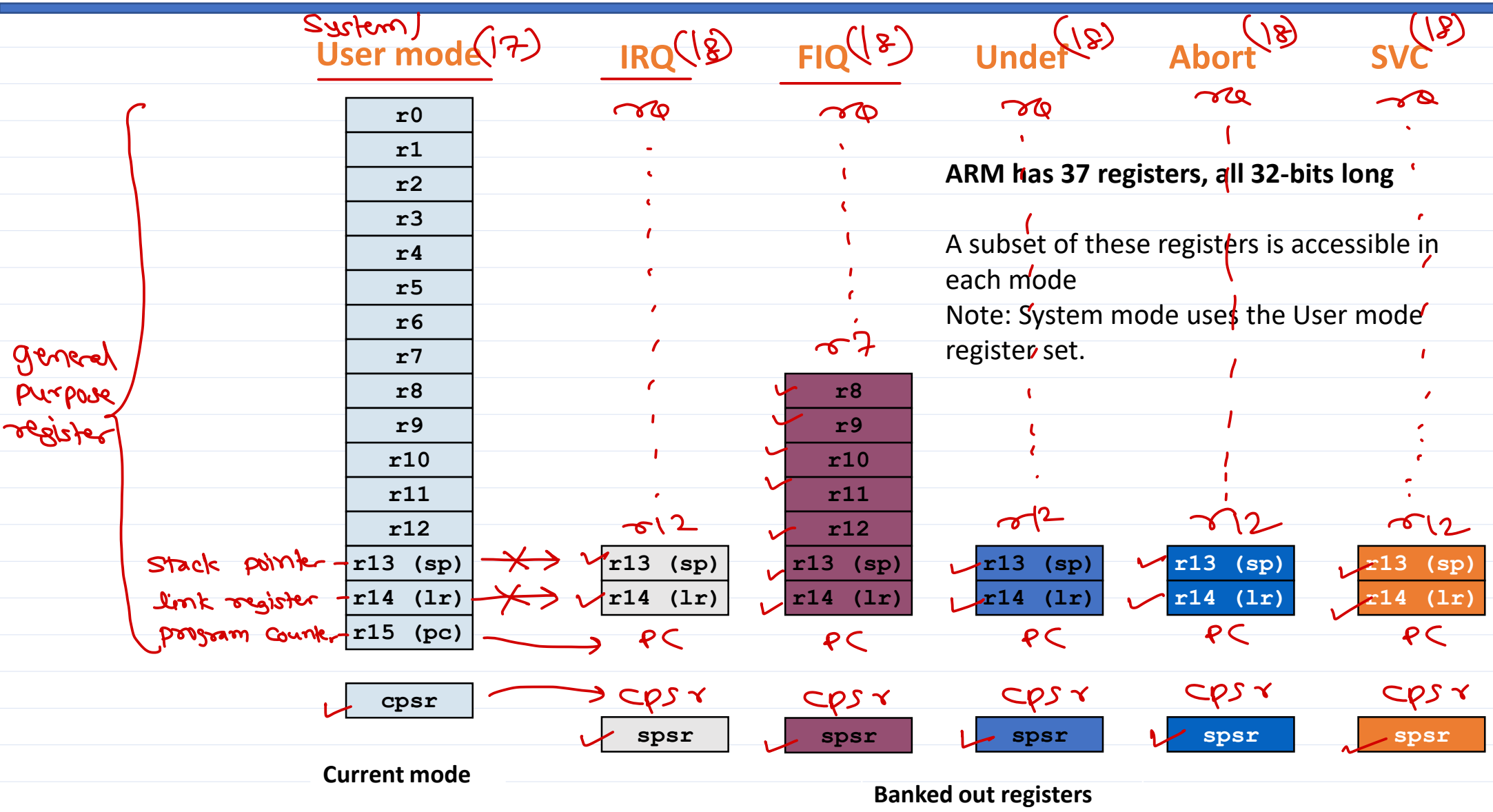
Mode	Description	
Supervisor (SVC)	Entered on reset and when a Supervisor call (earlier SWI) instruction (SVC) is executed	**Privileged modes**
FIQ	Entered when a high priority (fast) interrupt is raised	
IRQ	Entered when a normal priority interrupt is raised	
Abort	Used to handle memory access violations	
Undef	Used to handle undefined instructions	
System	Privileged mode using the same registers as User mode	
User	Mode under which most Applications / OS tasks run	**Unprivileged mode**

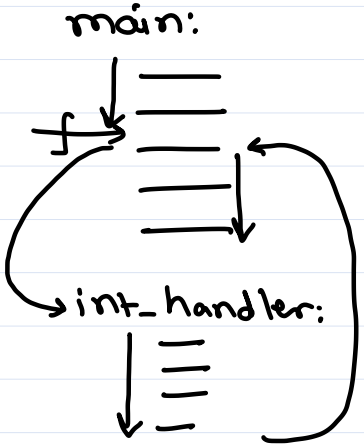
SysCalls are fns exposed by the kernel so that user programs can access kernel functionality.

→ Data processing, Control/Jump, Status regr manip, ...

→ Data processing, Control/Jump, ...

The ARM Register Set





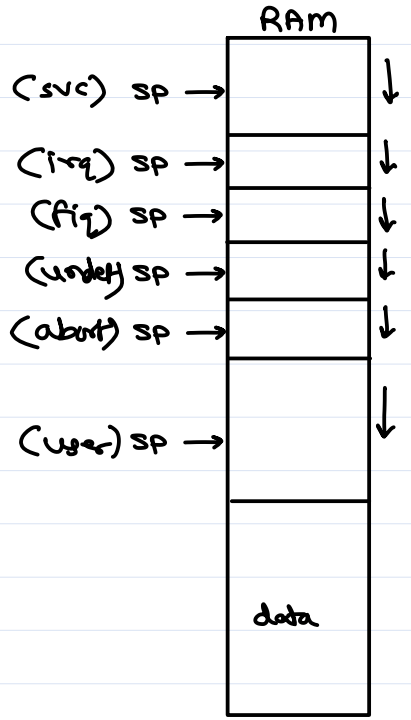
ARM has 37 registers, all 32-bits long

A subset of these registers is accessible in each mode

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 (cpsr)
  - 5 dedicated saved program status registers (spsr)
  - 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 Counter (r15)

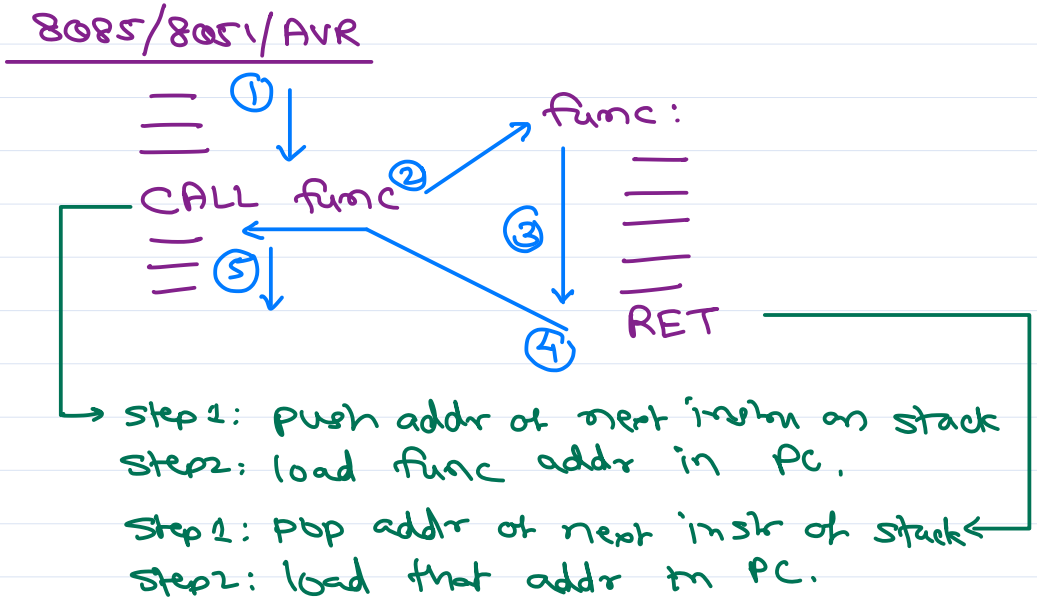
- When the processor is executing in ARM state:
  - All instructions are 32 bits wide
  - All instructions must be word aligned (address multiple of 4)
  - Therefore the **pc** value is stored in bits [31:2] with bits [1:0] undefined (as instruction cannot be halfword or byte aligned).

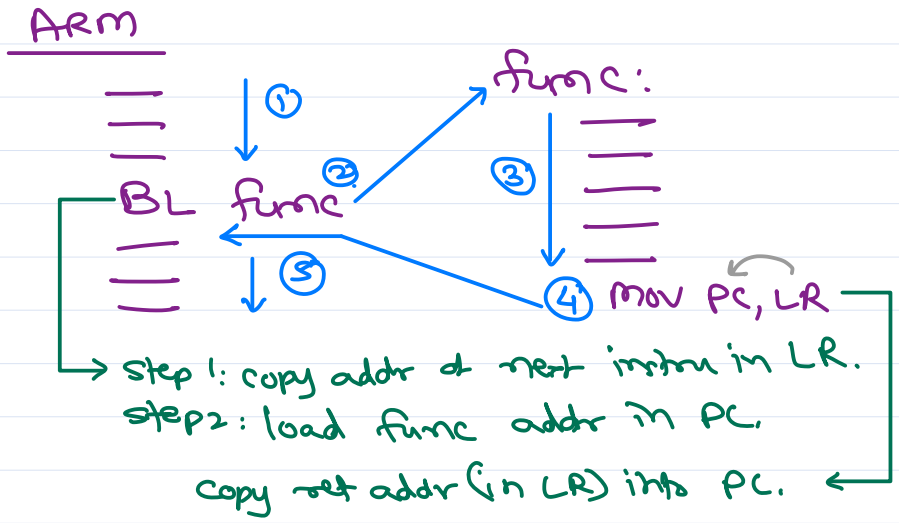
- When the processor is executing in Thumb state:
  - All instructions are 16 bits wide
  - All instructions must be halfword aligned
  - Therefore the **pc** value is stored in bits [31:1] with bit [0] undefined (as instruction cannot be byte aligned).

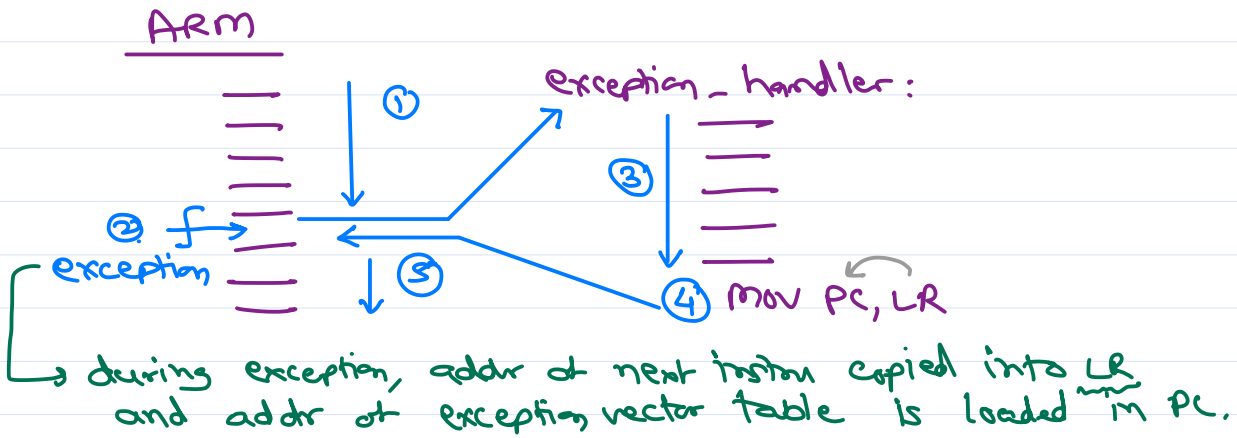
- When the processor is executing in Jazelle state:
  - All instructions are 8 bits wide
  - Processor performs a word access to read 4 instructions at once

LR revision

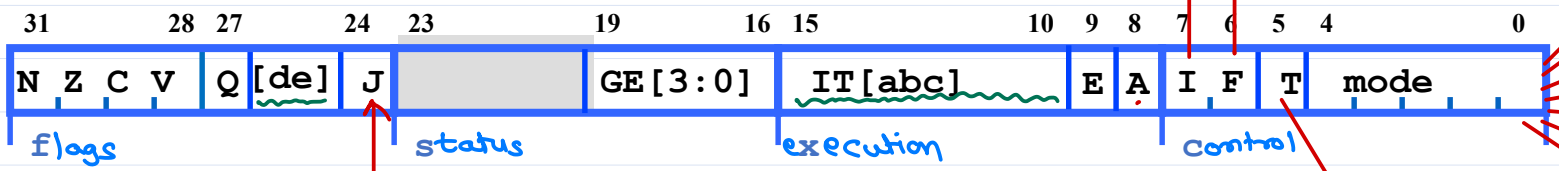
LR → link register.







Program Status Registers



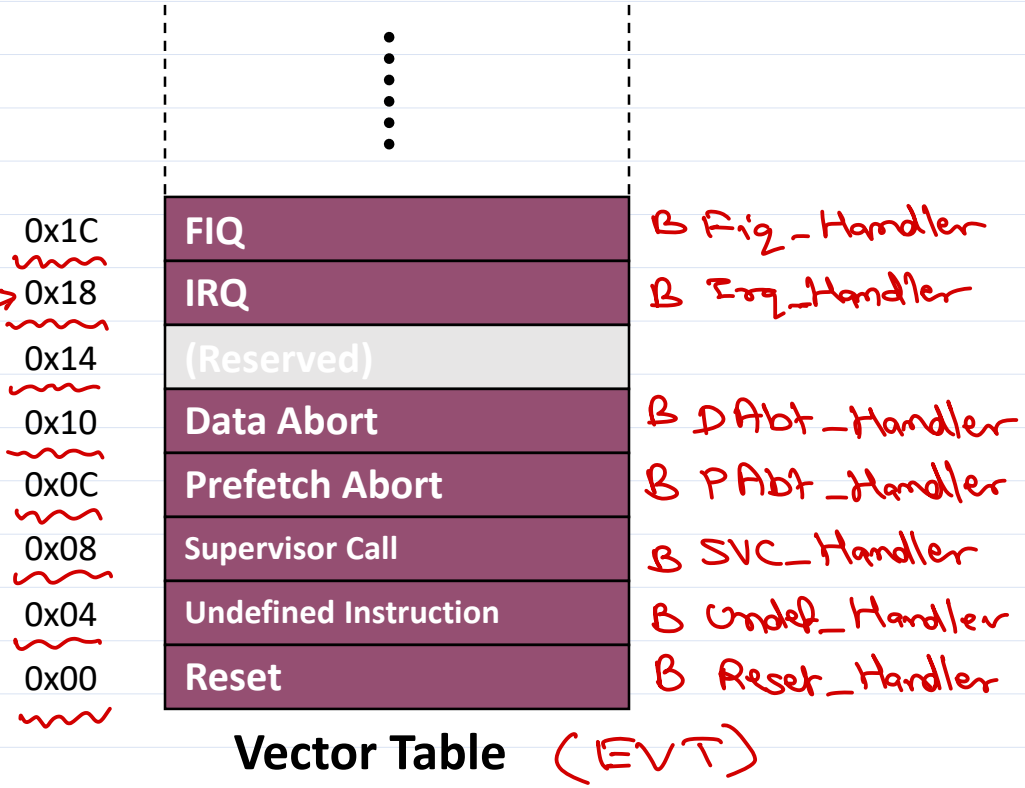
- Condition code flags
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation oVerflowed
- Sticky Overflow flag - Q flag
  - Indicates if saturation has occurred
- SIMD Condition code bits – GE[3:0]
  - Used by some SIMD instructions
- IF THEN status bits – IT[abcde]
  - Controls conditional execution of Thumb instructions
- T bit
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state
- J bit
  - J = 1: Processor in Jazelle state
- Mode bits
  - Specify the processor mode
- Interrupt Disable bits
  - I = 1: Disables IRQ
  - F = 1: Disables FIQ
- E bit
  - E = 0: Data load/store is little endian
  - E = 1: Data load/store is bigendian
- A bit
  - A = 1: Disable imprecise data aborts

Exception Handling

- When an exception occurs, the core…
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state (if appropriate)
    - 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>

- Cores can enter ARM state or Thumb state when taking an exception
  - Controlled through settings in CP15

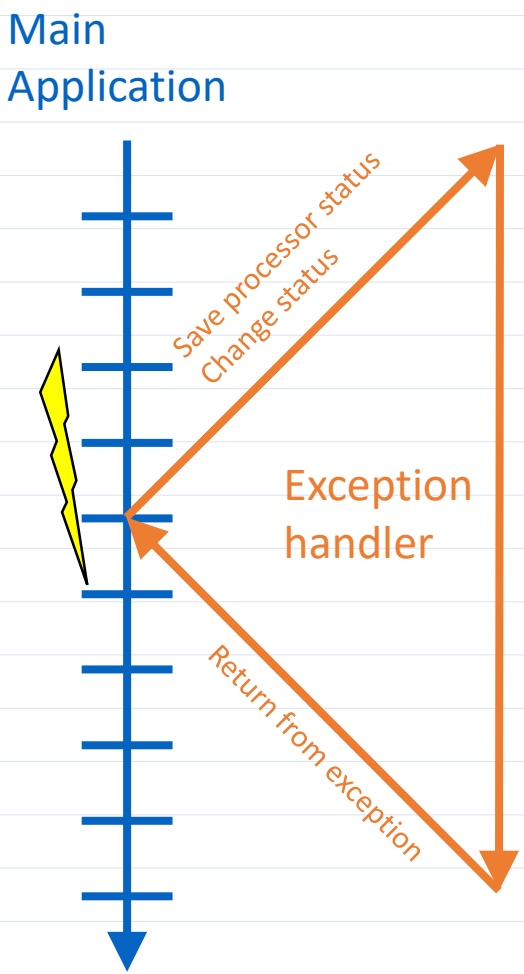
- Note that v7-M and v6-M exception model is different



Vector Table

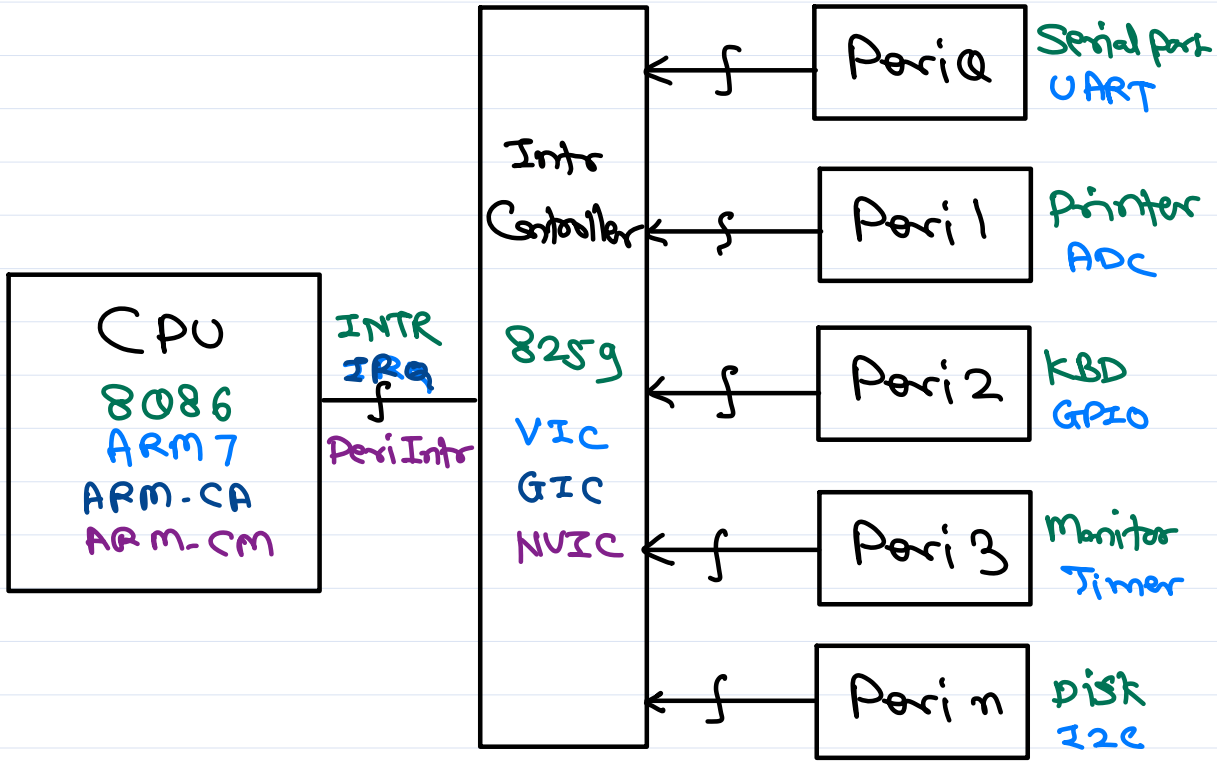
Vector table can also be at **0xFFFF0000** on most cores

Exception handling process



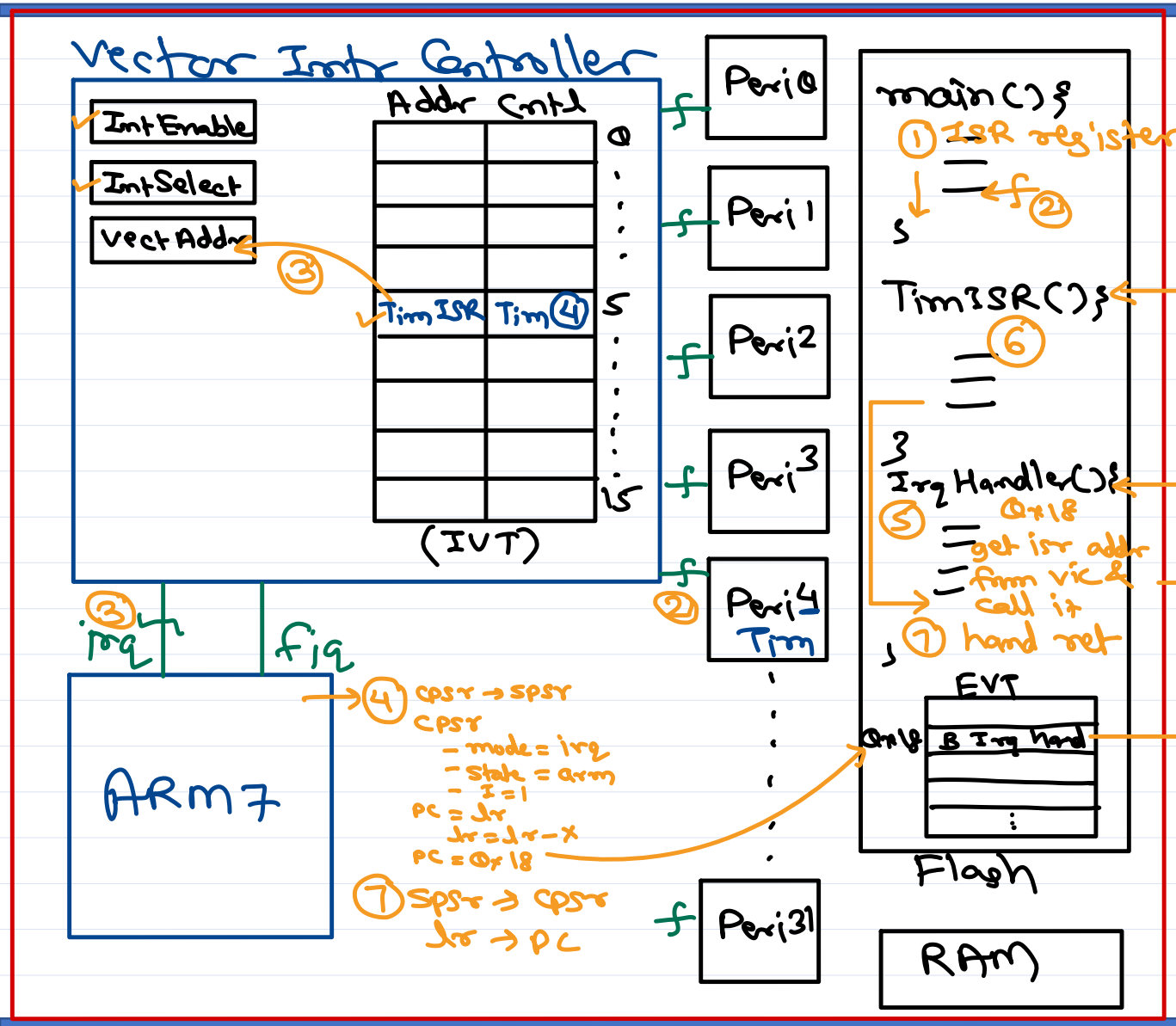
1. Save processor status
   - Copies `CPSR` into `SPSR_<mode>`
   - Stores the return address in `LR_<mode>`
   - Adjusts LR based on exception type LR = LR - 4 or LR = LR - 8
2. Change processor status for exception
   - Mode field bits
   - ARM or Thumb state
   - Interrupt disable bits (if appropriate) I/F
   - Sets PC to vector address
3. Execute exception handler
   - <users code>
4. Return to main application
   - Restore `CPSR` from `SPSR_<mode>`
   - Restore PC from `LR_<mode>`

- 1 and 2 performed automatically by the core
- 3 and 4 responsibility of software


CPU
8086
ARM7
ARM-CA
ARM-CM
INTR
IRQ
PeriIntr
Intr
Controller
8259
VIC
GIC
NVIC
Peri0
Peri1
Peri2
Peri3
Peri n
Serial port
UART
Printer
ADC
KBD
GPIO
Monitor
Timer
Disk
I2C

ARM7 core interrupt handling

LPC2148



irq_handler:

1. Save current exec context on stack.
2. get ISR addr from VIC.
3. invoke ISR.
4. restore exec context from stack.

By default, when IRQ intr arrives, IRQ is disabled (I=1). Irq_handler may reenable intr (I=0) so that nested intrs are allowed.

IRQ intr latency = 24-30 cycles

IRQ vs FIQ

1. FIQ has higher prio than IRQ.
2. FIQ has a set GPR (r8-r12) to be used in handler. So no need to save & restore exec. ctx.
3. FIQ vector addr is last in EVT. Instead of jump instn directly handler can be implemented there.
4. Usually only one peri is configured as FIQ. So handler can directly start handling intr (instead of getting info from VIC).

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>