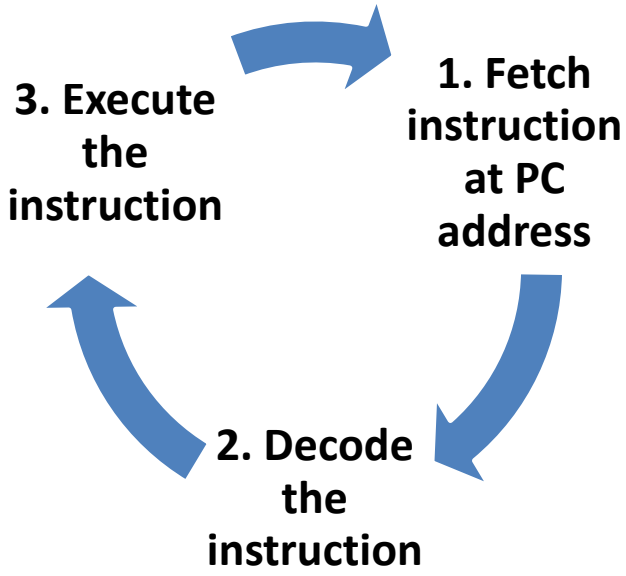
Exceptions

- Exceptions – events outside normal flow of execution



- None of our programs so far have deviated from the fetch-decode-execute cycle
- None of our programs so far have interacted with the outside world (external devices)

Examples of exceptions

- Expected events that occur at unpredictable times (with respect to the execution of our program!)
  - Mouse clicks, keyboard presses, touchscreen presses, button presses, ...
  - Receive network data, finish sending network data, ...
  - Wait for a timer to count down to zero, ...
- Unexpected events that we can try to handle
  - Eject CD, remove USB key, ...
  - Battery power low, ...
  - Loose wireless network signal, network cable unplugged, ...
  - Read from or write to invalid memory addresses
  - Attempting to execute invalid machine code
  - Reset

Polling or Interrupts

- Suppose a memory location (e.g. 0xE1000000) is set to the value 1 when a push-button is pressed
- Want to write a program to take some action when the button is pressed
- Approach 1
  - keep reading 0xE1000000 until we read 0
  - do nothing between tests
- Approach 2
  - keep reading 0xE1000000 until we read 0
  - do useful things between tests

"polling"

- Approach 3
  - do useful things
  - break out of F-D-E cycle when event occurs

"interrupts"

ARM Exceptions

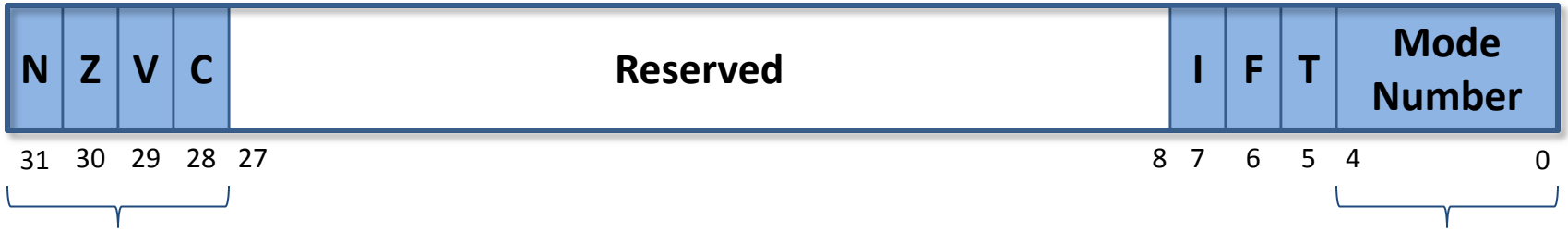
Name	Description
Reset	Occurs at power-on or when RESET button is pressed
Undefined Instruction	Attempt to execute an invalid instruction word
Software Interrupt (SWI)	Caused programmatically by executing SWI instruction
Prefetch Abort	Attempt to fetch an instruction from an invalid address (instructions)
Data Abort	Attempt to load/store from/to an invalid address (data)
(Reserved)	
IRQ	Iterrupt ReQuest
FIQ	Fast Interrupt reQuest

When one of these executions occurs, it needs to be **"handled"** by an **"exception handler"** (like a subroutine) that takes appropriate action.

ARM Programmers' Model – Modes

Processor mode		Mode number	Description
User	usr	0b10000	Normal program execution
FIQ	fiq	0b10001	Fast Interrupt reQuest handling (low overhead)
IRQ	irq	0b10010	General purpose interrupt handling
Supervisor	svc	0b10011	Protected mode for OS (Reset / SWI)
Abort	abt	0b10111	Prefetch / Data Abort (memory management)
Undefined	und	0b11011	Software emulation of unimplemented instructions
System	sys	0b11111	Privileged mode using user-mode registers (OS)

Current Program Status Register



Condition Code Flags

Processor Mode

ARM Programmers' Model – Registers

ARM Architecture Reference Manual (Issue I), Figure A2-1

Modes						
	Privileged modes					
		Exception modes				
User	**System**	**Supervisor**	**Abort**	**Undefined**	**Interrupt**	**Fast interrupt**
R0	R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7	R7
R8	R8	R8	R8	R8	R8	R8_fiq
R9	R9	R9	R9	R9	R9	R9_fiq
R10	R10	R10	R10	R10	R10	R10_fiq
R11	R11	R11	R11	R11	R11	R11_fiq
R12	R12	R12	R12	R12	R12	R12_fiq
R13	R13	R13_svc	R13_abt	R13_und	R13_irq	R13_fiq
R14	R14	R14_svc	R14_abt	R14_und	R14_irq	R14_fiq
PC	PC	PC	PC	PC	PC	PC
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_svc	SPSR_abt	SPSR_und	SPSR_irq	SPSR_fiq

indicates that the normal register used by User or System mode has been replaced by an alternative register specific to the exception mode

Exception handling sequence

(For ARM7TDMI and all ARM versions up to v7 (Cortex))

1. Complete execution of current instruction
2. Save current state and update CPSR
   - CPSR saved to SPSR_<mode> where <mode> will be the new processor mode while the exception is being handled
   - Switch to ARM state (clear T bit)
   - Disable IRQs (set I bit)
   - Disable FIQs (set F bit) when handling FIQs and Reset
   - Save return address (address of next instruction) in LR_<mode>, where <mode> will be the new processor mode while handling the exception
3. Change PC to address of the **exception handler** subroutine corresponding to the exception type

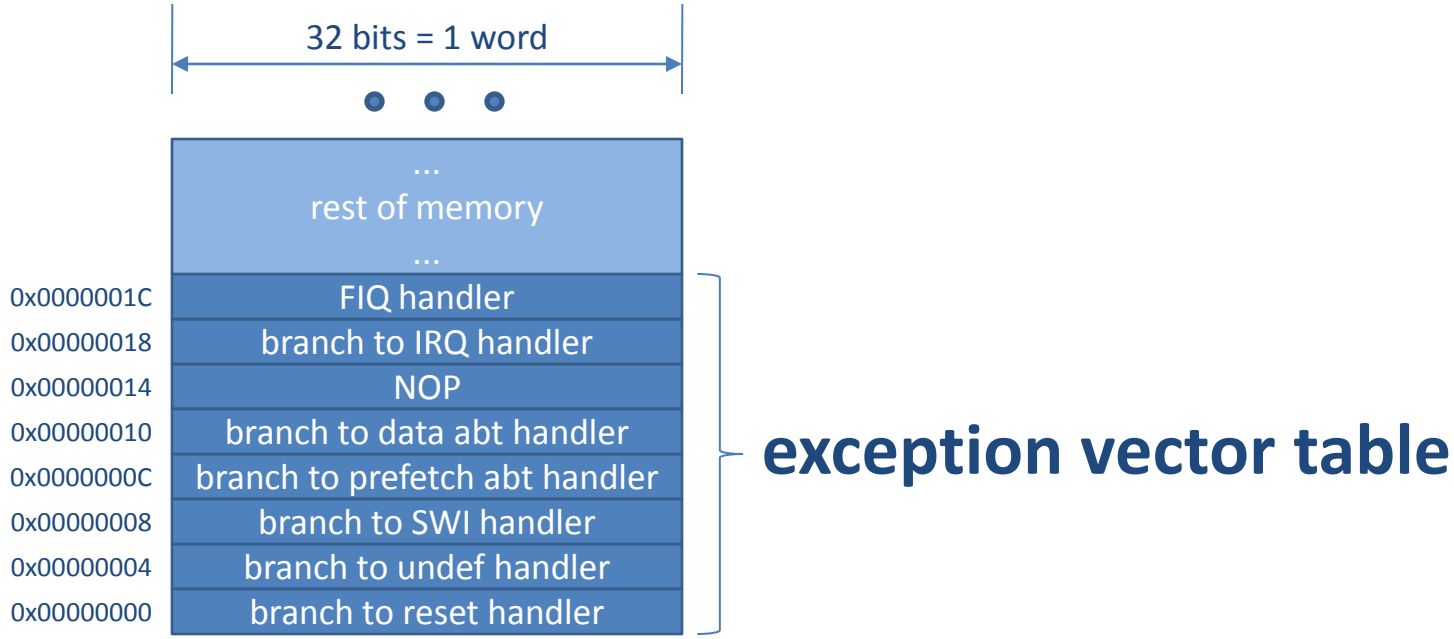
Exception vector table

- Start address for the exception hander for each exception type is fixed and well known
- Processor loads PC with this fixed, well-known address

Exception Type	Handler Start Address
Reset	0x00000000
Undefined Instruction	0x00000004
Software Interrupt (SWI)	0x00000008
Prefetch Abort	0x0000000C
Data Abort	0x00000010
(Reserved)	0x00000014
IRQ	0x00000018
FIQ	0x0000001C

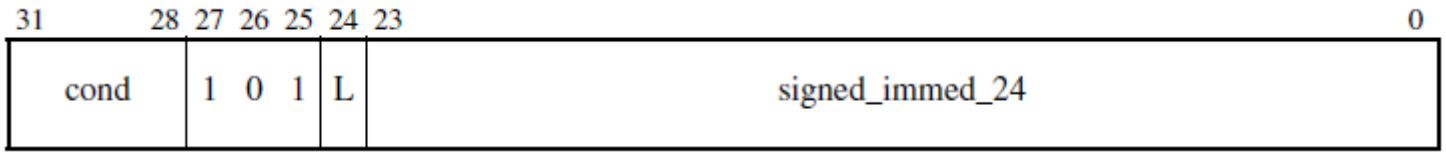
Exception vector table

- Obviously cannot store much of a subroutine in the one word available for each exception type!!
- Instead, each of the eight words at the start of memory contains an instruction that causes a branch to the real exception handler for the exception type



Exception vector table

- A number of ways we can branch to the start of the ***real*** exception handler ...
- A branch (B) instruction – range limited to 32MB



- A MOV instruction – limited to jumping to an address that can be represented as a byte shifted by an even number of bits
- An LDR instruction that loads any address from a PC-relative location in memory

```
LDR      PC, [PC + offset]
```

Exception vector table

```
                AREA    RESET, CODE, READONLY

; Exception Vectors
; Mapped to Address 0.
; Absolute addressing mode must be used.
; Dummy Handlers are implemented as infinite loops which can be modified.

Vectors         LDR     PC, Reset_Addr
                LDR     PC, Undef_Addr
                LDR     PC, SWI_Addr
                LDR     PC, PAbt_Addr
                LDR     PC, DAbt_Addr
                NOP
                LDR     PC, [PC, #-0x0120]
                LDR     PC, FIQ_Addr

Reset_Addr      DCD     Reset_Handler
Undef_Addr      DCD     Undef_Handler
SWI_Addr        DCD     SWI_Handler
PAbt_Addr       DCD     PAbt_Handler
DAbt_Addr       DCD     DAbt_Handler
                DCD
                DCD
FIQ_Addr        DCD     FIQ_Handler
                ...     ...
```

exception vector table

addresses of real exception handlers (“literal pool”)

Exception vector table

```
                ...     ...

Undef_Handler   B       Undef_Handler
SWI_Handler     B       SWI_Handler
PAbt_Handler    B       PAbt_Handler
DAbt_Handler    B       DAbt_Handler
IRQ_Handler     B       IRQ_Handler
FIQ_Handler     B       FIQ_Handler


; Reset Handler

                EXPORT  Reset_Handler
Reset_Handler
                < reset handler code goes here >
```

dummy default exception handlers

real reset handler

- IRQ handler address is specified differently
- FIQ is a special case – designed to reduce overheads and allow faster exception handling

Exception priorities

- Multiple exceptions can happen at the same time
- or, an exception might happen while another exception is already being handled
- Exceptions are prioritised

Exception Type	Priority
Reset	Highest
Data Abort	
FIQ	
IRQ	
Prefetch Abort	
Software Interrupt (SWI)	
Undefined Instruction	Lowest

- Lower priority exceptions are only handled after the handler for higher priority exceptions has completed

Example: reset exceptions

- Begin fetch from 0x00000000 when processor is powered or reset, causing execution of reset handler
- Reset handler can
  - Set up exception vectors to override defaults
  - Initialise memory controller for memory devices
  - Initialise stacks (SPs) for each processor mode
  - Initialise I/O devices
  - Initialise peripheral devices, including clocks
  - Enable interrupts
  - Change the processor mode
  - Branch to a start-up program

Example: undefined instruction exceptions

- ARM instructions are 32-bits long
  - 2^{32} possible instruction words
  - not all instruction words are valid
  - invalid instructions raise undefined instruction exceptions
- Take advantage of this to extend the instruction with our own instructions
  - Instruction operation must be implemented in software
  - When the processor attempts to execute it, an undefined instruction exception is raised
  - Undefined instruction exception handler is executed
  - We provide our own undefined instruction exception handler to decode the instruction and implement the desired operation

Example: Undefined POWER instruction

- Provide a POWER instruction to compute x^y
- Define our instruction template

POWER instruction

cond	0111	1111	opcode	Rn	Rd	1111	Rm

- Write our Undefined exception handler
- Set up the vector table
- Test the instruction

Example: Undefined POWER instruction

```
;
; Undefined exception handler
;
UndefHandler
            STMFD    sp!, {r0-r12, LR}          ; save registers

            LDR      r4, [lr, #-4]              ; load undefinied instruction
            BIC      r5, r4, #0xFFF0FFFF        ; clear all but opcode bits
            TEQ      r5, #0x00010000            ; check for undefined opcode 0x1
            BNE      endif1                     ; if (power instruction) {

            BIC      r5, r4, #0xFFFFFFF0        ;  isolate Rm register number
            BIC      r6, r4, #0xFFFF0FFF        ;  isolate Rn register number
            MOV      r6, r6, LSR #12            ;
            BIC      r7, r4, #0xFFFFF0FF        ;  isolate Rd register number
            MOV      r7, r7, LSR #8             ;

            LDR      r1, [sp, r5, LSL #2]       ;  grab saved Rm off stack
            LDR      r2, [sp, r6, LSL #2]       ;  grab saved Rn off stack

            BL       power                      ;  call pow subroutine

            STR      r0, [sp, r7, LSL #2]       ;  save result over saved Rd
endif1                                          ; }
            LDMFD    sp!, {r0-r12, PC}^         ; restore register and CPSR
```

Example: Undefined POWER instruction

```
            AREA     Undef, CODE, READONLY
            IMPORT   main
            EXPORT   start

start
            LDR      r4, =0x40000024  ; 0x40000024 is mapped to 0x00000024
            LDR      r5, =UndefHandler ; Address of new undefined handler
            STR      r5, [r4]         ; Store new undef handler address

            ;
            ; Test our new instruction
            ;
            LDR      r4, =3           ; test 3^4
            LDR      r5, =4           ;

            ; This is our undefinied unstruction opcode
            DCD      0x77F150F4       ; POW r0, r4, r5 (r0 = r4 ^ r5)

            ; R0 should be 81

stop        B        stop
```

Example: Undefined POWER instruction

```
; power subroutine
; Computes x^y
; paramaters: r0: result (variable)
;             r1: x (value)
;             r2: y (value)
power
            STMFD    sp!, {r1-r2,lr}  ; save registers

            CMP      r2, #0                          ; if (y = 0)
            BNE      else2                           ; {
            MOV      r0, #1                          ;  result = 1
            B        endif2                          ; }
else2                                                ; else {
            MOV      r0, r1                          ;  result = x
            SUBS     r2, r2, #1                      ;  y = y - 1
            BEQ      endif3                          ;  if (y != 0) {
do4                                                  ;   do {
            MUL      r0, r1, r0                      ;    result = result * x
            SUBS     r2, r2, #1                      ;    y = y - 1
            BNE      do4                             ;   } while (y != 0)
endif3                                               ;  }
endif2                                               ; }

            LDMFD    sp!, {r1-r2, pc} ; restore registers and return
```