

Programmer's Model (1A)

Copyright (c) 2014 - 2018 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using LibreOffice.

Based on

ARM System-on-Chip Architecture, 2nd ed, Steve Furber

Architectural Inheritance

A load-store architecture

Fixed-length 32-bit instructions

3-address instruction formats

ARM Instruction Set

The load-store architecture

3-address data processing instructions

(2 source registers + 1 destination register)

Conditionally executes every instruction

Multiple data transfer instruction

Single cycle execution of shift and ALU operations

Open instruction set for coprocessors

A very dense 16-bit compressed instruction set (Thumb)

ARM Exceptions

Interrupts

Traps

Supervisor calls

ARM Exception Types

- Reset
- Non-maskable Interrupt
- Hard Fault
- Memory Management
- Bus Fault
- Usage Fault
- SVCall
- Debug Monitor
- PendSV
- Sys Tick
- External Interrupt

ARM Processor Operating Modes

- User – normal mode for most programs (tasks)
- FIQ (Fast Interrupt)
- IRQ (Interrupt)
- SVC (Supervisor)
- ABT (Abort)
- UND (Undefined)
- System – use the same registers as User mode

ARM Processor Normal Registers

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)

CPSR

- $16 + 1 = 17$ normal registers

Each mode accesses a subset of normal registers

User	System	Fast Interrupt	Interrupt	Supervisor	Abort	Undefined
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
R9	R9		R9	R9	R9	R9
R10	R10		R10	R10	R10	R10
R11	R11		R11	R11	R11	R11
R12	R12		R12	R12	R12	R12
R13 (SP)	R13 (SP)					
R14 (LR)	R14 (LR)					
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR

ARM Processor Alternative Registers

additional hardware registers

Registers for
Fast Interrupt

Registers for
Interrupt

Registers for
Supervisor

Registers for
Abort

Registers for
Undefined

R8_fig
R9_fig
R10_fig
R11_fig
R12_fig
R13_fig
R14_fig

- $8+3+3+3+3 = 20$ alternative registers

R13_irq
R14_irq

R13_svc
R14_svc

R13_abt
R14_abt

R13_und
R14_und

SPSR_fiq

SPSR_irq

SPSR_svc

SPSR_abt

SPSR_und

ARM Processor General Purpose Registers

R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13 (SP)
R14 (LR)
R15 (PC)

CPSR

- 37 general purpose registers
- 16 + 1 = 17 normal registers
- 8+3+3+3+3 = 20 alternative registers

R8_fig
R9_fig
R10_fig
R11_fig
R12_fig
R13_fig
R14_fig

R13_irq
R14_irq

R13_svc
R14_svc

R13_abt
R14_abt

R13_und
R14_und

SPSR_fig

SPSR_irq

SPSR_svc

SPSR_abt

SPSR_und

ARM Processor Registers

User	System	Fast Interrupt	Interrupt	Supervisor	Abort	Undefined
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_fig	R8	R8	R8	R8
R9	R9	R9_fig	R9	R9	R9	R9
R10	R10	R10_fig	R10	R10	R10	R10
R11	R11	R11_fig	R11	R11	R11	R11
R12	R12	R12_fig	R12	R12	R12	R12
R13 (SP)	R13 (SP)	R13_fig	R13_irq	R13_svc	R13_abt	R13_und
R14 (LR)	R14 (LR)	R14_fig	R14_irq	R14_svc	R14_abt	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR_fiq	SPSR_irq	SPSR_svc	SPSR_abt	SPSR_und

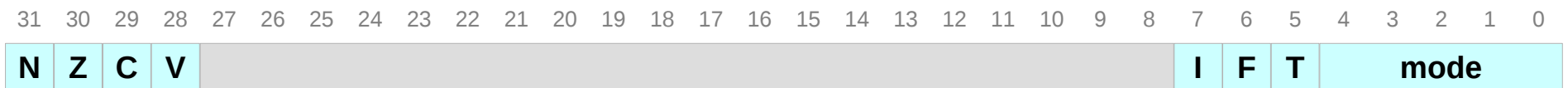
<http://www.cs.otago.ac.nz/cosc440/readings/arm-syscall.pdf>

ARM Exception Handling

1. Save the current state
 - copying the **PC** into **r14_exec**
 - copying the **CPSR** into **SPSR_exec**
 - exec : exception type
2. changing the processor operating mode
 - To the appropriate exception mode
3. **PC** is forced to have values between **0x00** and **0x1C**
 - depending on the type of exception
 - exception handlers

CPSR – ALU flags

- N** Negative flag – the last ALU operation yields a negative result
- Z** Zero flag – the last ALU operation yields a zero result
- C** Carry flag – the last ALU operation yields a carry out
- V** Overflow flag – the last ALU operation yields a overflowed result

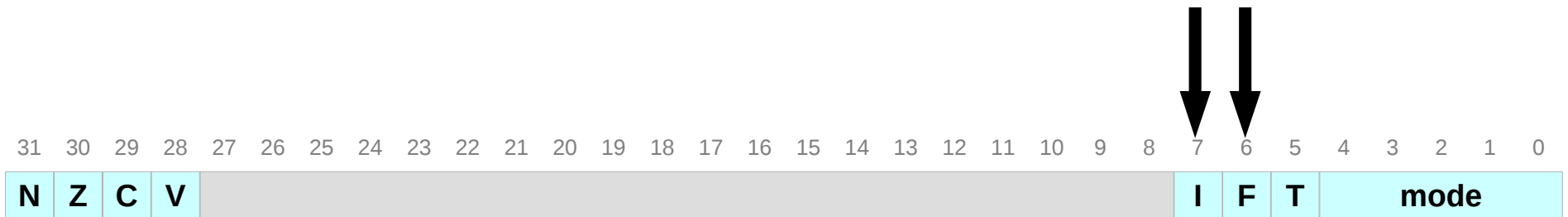


Current Program Status Register (CPSR)

CPSR – Interrupt Disable Bits

To disable Interrupt (IRQ), set **I**

To disable Fast Interrupt (FIQ), set **F**



Current Program Status Register (CPSR)

CPSR – ARM / Thumb Indicator

the **T** bit shows whether the processor runs in ARM state or in Thumb state.

never set this bit

can be changed only in a privileged mode



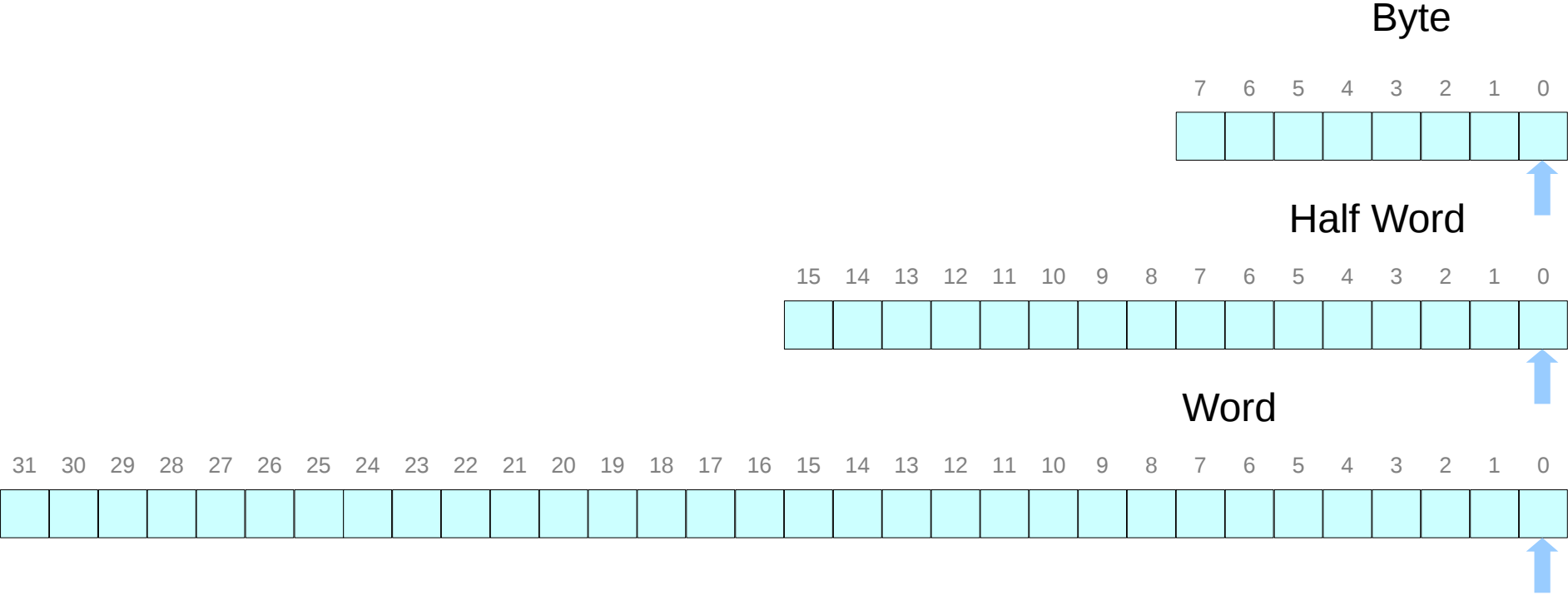
Current Program Status Register (CPSR)

CPSR – Mode Indicator

Usr (usr)	1	0	0	0	0
Fast Interrupt (fiq)	1	0	0	0	1
Interrupt (irq)	1	0	0	1	0
Supervisor (svc)	1	0	0	1	1
Abort (abt)	1	0	1	1	1
Undefined (und)	1	1	0	1	1
System (sys)	1	1	1	1	1



ARM Data Unit



ARM Byte Address

Little Endian Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte 31				Byte 30				Byte 29				Byte 28																			
Byte 27				Byte 26				Byte 25				Byte 24																			
Byte 23				Byte 22				Byte 21				Byte 20																			
Byte 19				Byte 18				Byte 17				Byte 16																			
Byte 15				Byte 14				Byte 13				Byte 12																			
Byte 11				Byte 10				Byte 9				Byte 8																			
Byte 7				Byte 6				Byte 5				Byte 4																			
Byte 3				Byte 2				Byte 1				Byte 0																			

increasing address

ARM Byte Address Examples

Little Endian Mode

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Byte at 31								Byte at 30								Byte at 29								Byte at 28							
Byte at 27							Byte at 26							Byte at 25							Byte at 24										
Byte at 23							Byte at 22							Byte at 21							Byte at 20										
Word at 16																															
Half Word at 14															Half Word at 12																
Word at 8																															
Byte at 7							Byte at 6							Half Word at 4																	
Byte at 3							Byte at 2							Byte at 1							Byte at 0										

byte address of the least significant byte

Load-Store Architecture

No memory-to-memory operations

1. Data Processing Instructions
2. Data Transfer Instructions
3. Control Flow Instructions

Load-Store Architecture

No memory-to-memory operations

1. Data Processing Instructions

Change only register values

2. Data Transfer Instructions

Copy memory values into registers (**load**)

Copy register values into memory (**store**)

Exchange a memory value with a register value

3. Control Flow Instructions

Causes execution to switch to a different address

Permanently (**branch**)

Saving return address to resume (**branch** and **link**)

Trap into system code (**supervisor call**)

ARM Instruction Set

The load-store architecture

3-address data processing instruction

Conditional execution of every instruction

Very powerful load and store multiple register instructions

Single cycle general shift operation

Single cycle general ALU operation

Open instruction set

- co-processor instruction set

- Adding new registers and data types

A very dense 16-bit compressed thumb instruction set

ARM IO System

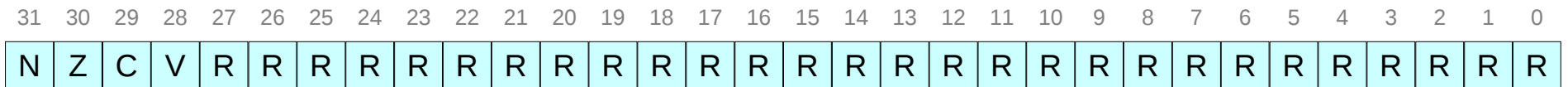
Memory mapped devices with interrupt support

Read / write the internal registers of these devices

Using the same load and store instructions

Level sensitive and maskable interrupt (both IRQ and FIQ)

ARM Exception Handling



References

- [1] <ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf>
- [2] <https://www.umiacs.umd.edu/~hal/docs/daume02yaht.pdf>