

## PSR Register - flag bits

- Flag bits shows the results of the immediately preceding ALU calculation
  - N bit : Negative flag
    - ALU calculation generates negative values
  - Z bit : Zero flag
    - ALU calculation generates zero
  - C bit : Carry flag
    - ALU calculation or shift operation generates carry
  - V bit : Overflow bit
- Additional bits
  - Q (Saturation bit), J (Jazelle state) bits

## PSR Register - control bits

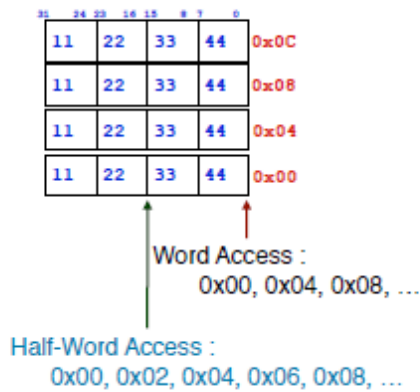
- PSR control bits change processor operation modes, interrupts, and other processor states
  - I/F bit
    - enable/disable IRQ or FIQ
  - T bit
    - Thumb mode on/off
    - Only controlled through 'BX' instruction
  - Mode bits

M[4:0]	Operating Mode	M[4:0]	Operating Mode
10000	User	10111	Abort
10001	FIQ	11011	Undefined
10010	IRQ	11111	System
10011	SVC		

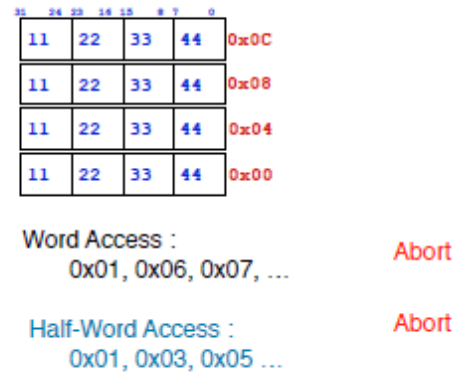
## Memory Architecture

- Endian
  - Big / Little Endian (ARM supports both)
  - Configured through hardware setting
- Data access types
  - Byte : 8 bits
  - Halfword : 16 bits
  - Word : 32 bits (in 32 bits processors)
- Aligned/Un-aligned access

### Aligned Access

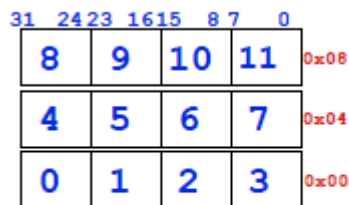


### Un-aligned Access

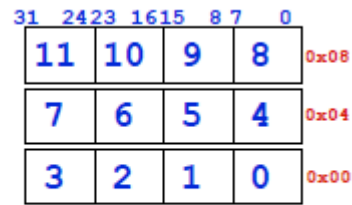


- All the memory accesses should be aligned by word(half-word)
- Modern architecture allows un-aligned accesses
  - previously always data aborts occur

## Little-Endian VS Big-Endian



### Big-Endian



### Little-Endian

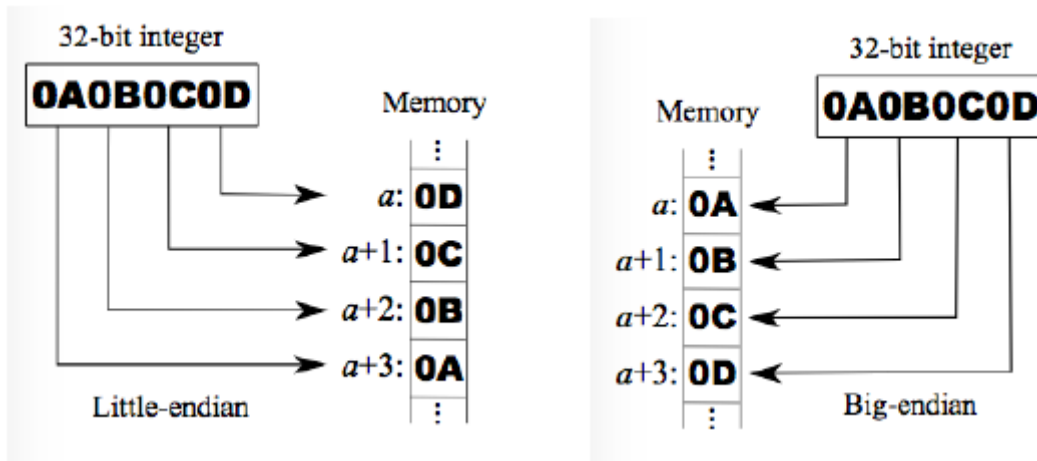
- Big-Endian
  - MSB is located in lower addresses in memory(Bits 24~31 are MSB)
- Little-Endian
  - LSB is located in lower addresses in memory(Bits 0 ~ 7 are LSB)

**Table 7.108. Little-endian write, 64-bit external bus**

Internal transfer width	Access: Write, little-endian, 64-bit external bus		System data mapping on to external data bus MPMCDATA to HRDATA							
	H SIZE [2:0]	H ADDR [2:0]	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Byte	000	011	-	-	-	-	[31:24]	-	-	-
Byte	000	010	-	-	-	-	-	[23:16]	-	-
Byte	000	001	-	-	-	-	-	-	[15:8]	-
Byte	000	000	-	-	-	-	-	-	-	[7:0]

**Table 7.109. Big-endian write, 64-bit external bus**

Internal transfer width	Access: Write, big-endian, 64-bit external bus		System data mapping on to external data bus MPMCDATA to HRDATA							
	H SIZE [2:0]	H ADDR [2:0]	[63:56]	[55:48]	[47:40]	[39:32]	[31:24]	[23:16]	[15:8]	[7:0]
Byte	000	011	-	-	-	-	-	-	-	[7:0]
Byte	000	010	-	-	-	-	-	-	[15:8]	-
Byte	000	001	-	-	-	-	-	[23:16]	-	-
Byte	000	000	-	-	-	-	[31:24]	-	-	-



## Exceptions

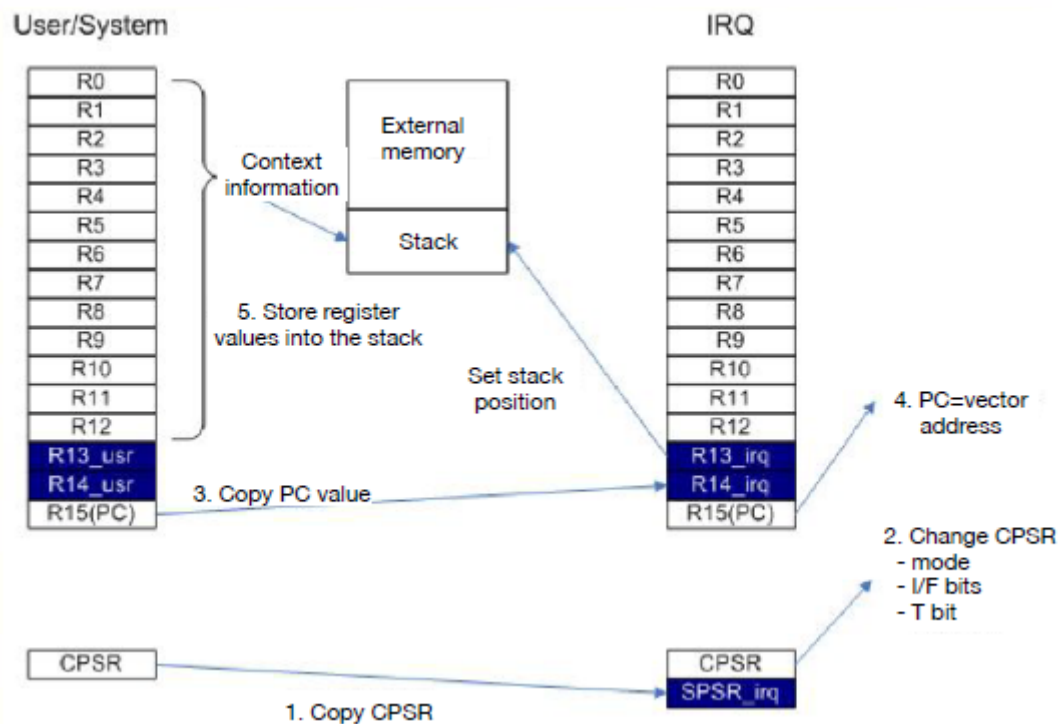
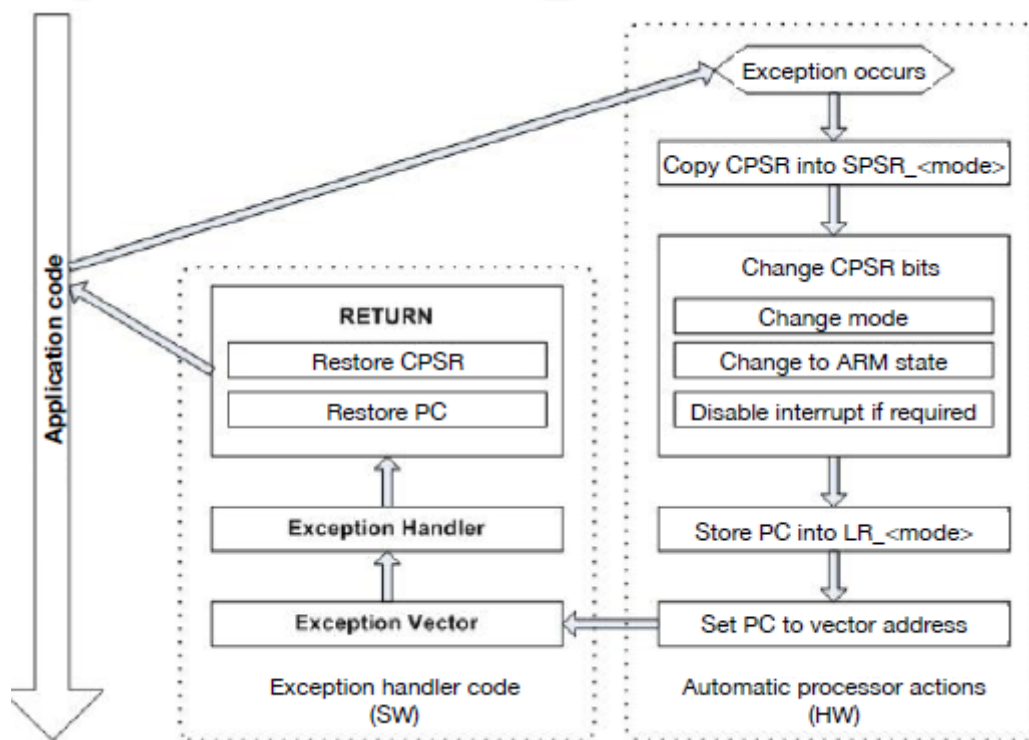
- Exceptions
  - Any interruption in regular control-flow execution caused by external requests, errors, and faults
  - Examples : external interrupts
- ARM exceptions
  - reset, undefined instruction, software interrupt, prefetch abort, data abort, IRQ, FIQ

## Exception Vectors

- Exception vector and Exception vector table
  - The pre-determined address to handle each exception
  - The table of exception handler addresses for all the exceptions
  - A fixed location in memory is allocated to exception vector table
  - Each entry of exception vector table is JUMP/BRANCH instruction to each exception handler
  - Default exception vector table location is 0x00000000

Exception	Vector Address	Priority	Operation mode
Reset	0x0000 0000	1 (High)	Supervisor(SVC)
Undefined Instruction	0x0000 0004	6 (Low)	Undefined
Software Interrupt(SWI)	0x0000 0008	6	Supervisor(SVC)
Prefetch Abort	0x0000 000C	5	Abort
Data Abort	0x0000 0010	2	Abort
Reserved	0x0000 0014		
IRQ	0x0000 0018	4	IRQ
FIQ	0x0000 001C	3	FIQ

## Exception Handling



## 32 Bits ARM Instruction Format

31	28 27	16 15	8 7
Cond	0 0 I	Opcode S Rn	Rd
Cond	0 0 0 0 0 0	A S Rd	Rn
Cond	0 0 0 0 1	U A S RdHi	RdLo
Cond	0 0 0 1 0	B 0 0 Rn	Rd
Cond	0 1 I	P U B W L Rn	Rd
Cond	1 0 0	P U S W L Rn	Register List
Cond	0 0 0	P U 1 W L Rn	Rd
Cond	0 0 0	P U 0 W L Rn	Rd
Cond	1 0 1	L Offset	
Cond	0 0 0 1 0 0 1 0	1 1 1 1	1 1 1 1
Cond	1 1 0	P U N W L Rn	CRd
Cond	1 1 1 0	Op1	CRn
Cond	1 1 1 0	Op1	L CRn
Cond	1 1 1 1	SWI Number	

## Conditional Execution

- Every instruction in ARM can be executed conditionally based on condition flag bits(previous ALU calculation)
- Significantly reduces branch costs causing pipeline stalls
- Condition fields
  - The condition fields of each instruction are compared with the condition flags in CPSR(Current Program Status Register)
  - The instruction can be executed only when the condition fields match condition flags in CPSR
- Condition to execute an instruction is described as postfix
  - Un-conditional execution

**ADD** r0,r1,r2 ; r0 = r1 + r2 (ADDAL)

- Conditional Execution

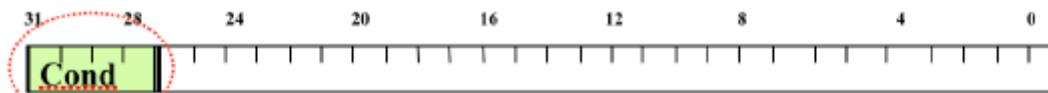
**ADDEQ** r0,r1,r2 ; If zero flag set then...  
; ... r0 = r1 + r2

- Data processing instructions with postfix 'S' change condition flags in CPSR

**SUBS** r0,r1,r2 ; r0 = r1 - r2  
; ... and set flags

- Comparison instructions always affect condition flags

## Condition Field



Opcode [31:28]	Mnemonic extension	Interpretation	Status flag state for execution
0000	EQ	Equal / equals zero	Z set
0001	NE	Not equal	Z clear
0010	CS/HS	Carry set / unsigned higher or same	C set
0011	CC/LO	Carry clear / unsigned lower	C clear
0100	MI	Minus / negative	N set
0101	PL	Plus / positive or zero	N clear
0110	VS	Overflow	V set
0111	VC	No overflow	V clear
1000	HI	Unsigned higher	C set and Z clear
1001	LS	Unsigned lower or same	C clear or Z set
1010	GE	Signed greater than or equal	N equals V
1011	LT	Signed less than	N is not equal to V
1100	GT	Signed greater than	Z clear and N equals V
1101	LE	Signed less than or equal	Z set or N is not equal to V
1110	AL	Always	any

## ARM Instruction Set Overview

---

Mnemonics	ARM ISA	Description
ADC	v1	add two 32-bit values and carry
ADD	v1	add two 32-bit values
AND	v1	logical bitwise AND of two 32-bit values
B	v1	branch relative $\pm 32$ MB
BIC	v1	logical bit clear (AND NOT) of two 32-bit values
BKPT	v5	breakpoint instructions
BL	v1	relative branch with link
BLX	v5	branch with link and exchange
BX	v4T	branch with exchange
CDP CDP2	v2 v5	coprocessor data processing operation
CLZ	v5	count leading zeros
CMN	v1	compare negative two 32-bit values
CMP	v1	compare two 32-bit values
EOR	v1	logical exclusive OR of two 32-bit values
LDC LDC2	v2 v5	load to coprocessor single or multiple 32-bit values
LDM	v1	load multiple 32-bit words from memory to ARM registers
LDR	v1 v4 v5E	load a single value from a virtual address in memory
MCR MCR2 MCRR	v2 v5 v5E	move to coprocessor from an ARM register or registers
MLA	v2	multiply and accumulate 32-bit values
MOV	v1	move a 32-bit value into a register
MRC MRC2 MRRC	v2 v5 v5E	move to ARM register or registers from a coprocessor
MRS	v3	move to ARM register from a status register ( <i>cpsr</i> or <i>spsr</i> )
MSR	v3	move to a status register ( <i>cpsr</i> or <i>spsr</i> ) from an ARM register
MUL	v2	multiply two 32-bit values
MVN	v1	move the logical NOT of 32-bit value into a register
ORR	v1	logical bitwise OR of two 32-bit values
PLD	v5E	preload hint instruction
QADD	v5E	signed saturated 32-bit add
QDADD	v5E	signed saturated double and 32-bit add
QDSUB	v5E	signed saturated double and 32-bit subtract
QSUB	v5E	signed saturated 32-bit subtract
RSB	v1	reverse subtract of two 32-bit values
RSC	v1	reverse subtract with carry of two 32-bit integers
SBC	v1	subtract with carry of two 32-bit values
SMLAxy	v5E	signed multiply accumulate instructions $((16 \times 16) + 32 = 32\text{-bit})$
SMLAL	v3M	signed multiply accumulate long $((32 \times 32) + 64 = 64\text{-bit})$
SMLALxy	v5E	signed multiply accumulate long $((16 \times 16) + 64 = 64\text{-bit})$
SMLAWy	v5E	signed multiply accumulate instruction $((32 \times 16) \gg 16) + 32 = 32\text{-bit})$
SMULL	v3M	signed multiply long $(32 \times 32 = 64\text{-bit})$