BITS Pilani presentation

BITS Pilani
Pilani Campus

Ramani Kalpathi
Automotive Electronics

innovate achieve lead

BITS Pilani

Pilani Campus

AEL ZG512 Embedded Systems
Lecture No. 3

Embedded Architecture 1 – RISC ARM Architecture

innovate achieve lead

Introduction to ARM CPU Architecture

Programmers Model of ARM CPU

- Register Organization
- Operating Modes
- Pipelining
- ARM Exception Handling

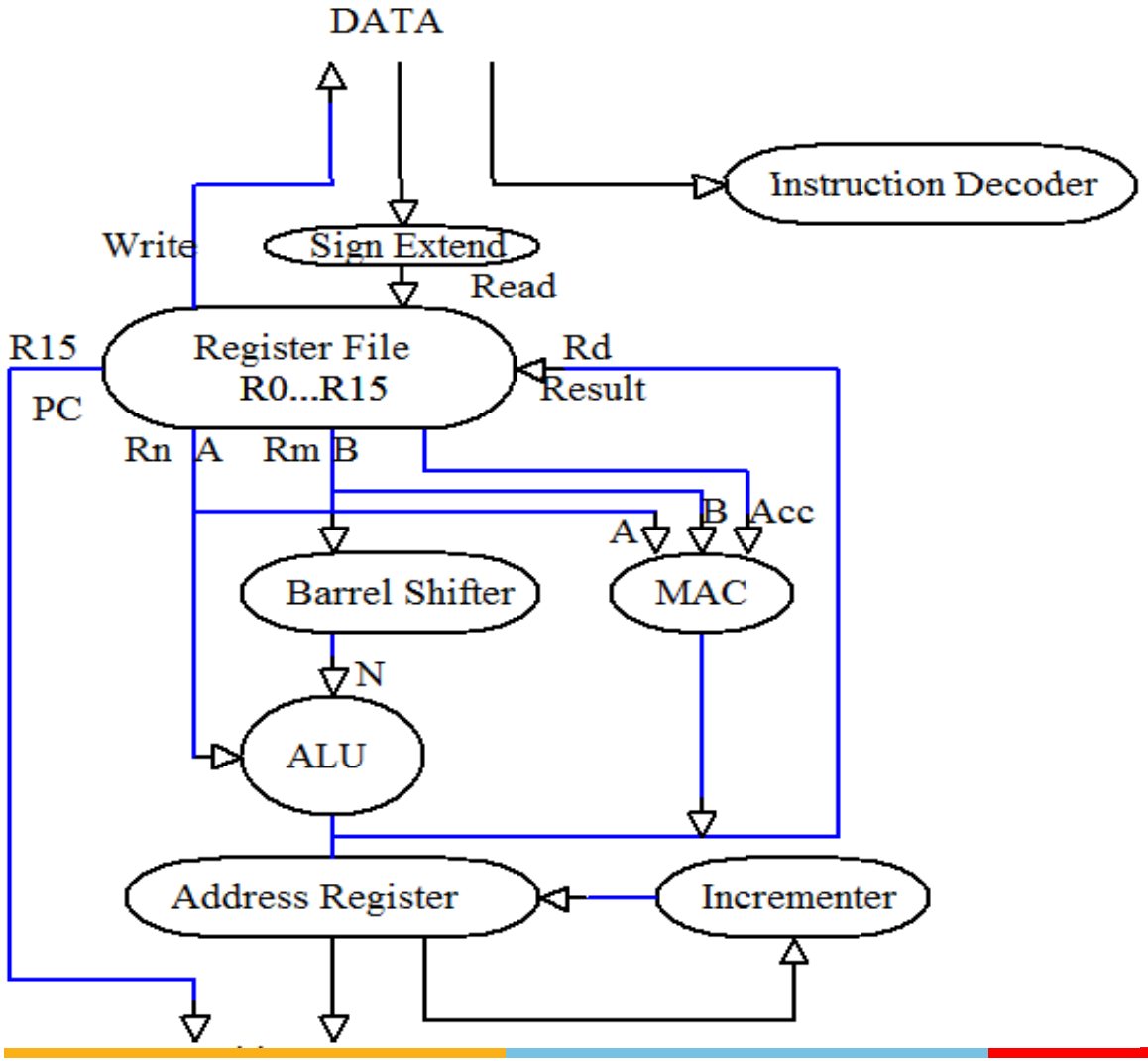
ARM Instruction Set

ARM CPU Architecture

In conventional RISC architectures, every instruction executes in a single cycle. The instruction set for the ARM architecture has the following variations.

- Some instructions take more than one cycle to complete
- Inline barrel shifter to assist in preprocessing input register data
- Instructions of 16-bit width - Thumb instruction set
- Instructions of 32-bit width – ARM instruction set
- Conditional instructions
- Enhanced instructions – multiplication capability for DSP

ARM core Data flow



Buses and Datapaths

- ARM core uses data paths or buses to transfer data between the computational units and the memory
- Data paths are used to move data between external memory and registers
- Data processing instructions can manipulate data in registers.
- The register file is a set of 32-bit memory banks. They can hold signed or unsigned data and have the ability to store data that are 8-bit, 16-bit or 32-bit wide that come from external devices through the data paths. A and B are the internal data buses and Rn, Rm are the source registers. Rd is the destination register.

ALU, Load Store mechanism

ALU refers to the arithmetic and logical unit.

MAC -multiply and accumulate unit. ALU and MAC compute result and write to the destination register Rd.

Load/Store instructions use ALU to generate an address to be held in the Address register and puts the address information on the address bus. The address information is required to fetch data from a memory address using load instruction or store information to the addressed memory.

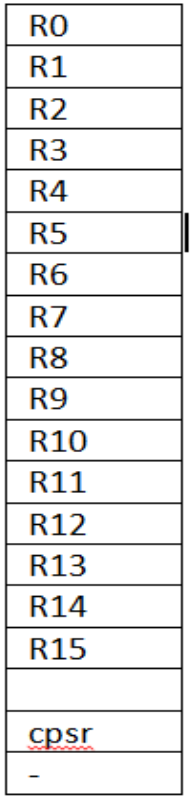
The barrel shifter is used to preprocess the address Rm before it enters the ALU.

The result bus is used to pass the result data of the computation.

The incrementer is used to point to the successive address.

Registers

- Registers are used to hold address or data.
- There are 18 registers that comprise the register bank
- Sixteen of them are data registers
- Two of them are status registers.
- Registers that can be accessed in user mode shown.
- Each register is 32 bits wide.
- R0-R12 are available for user programs
- R13-R15 and cpsr are available in protected mode



Registers..

- R13 is used as the stack pointer which stores the address of the top of the stack memory in the current processor mode.
- R14 is called the link register that stores the return address of the next instruction when the processor calls a subroutine.
- R15 is the program counter that stores the address of the next instruction to be fetched by the processor.
- Most instructions can be used on R0 to R12 while registers R13 and R14 have dedicated instructions. CPSR and SPSR are program status registers. CPSR is called the current program status register. SPSR is called the saved program status register.

Operating Modes and Program Status Register

31	30	29	28	27	26	25	24	23-16	15	14	13	12	11	10	9	8	7	6	5	4-0
N	Z	C	V														I	F	T	
Flags								Status	Extension								Interrupt masks			Mode

The current program status register is a 32 bit register called cpsr
Bits 0-7 are used for control
Bits 8-15 are for extension
Bits 16-23 for status
Bits 24-31 Flags that indicate the status of arithmetic or logical operations.

The control field contains the current processor mode, state of the processor (ARM or Thumb mode) and bits related to the interrupt mask bits.
The flags are related to condition flags.
The processor mode (Bits 0-4) determines access rights to the cpsr register
There are two types of processor modes, privileged and non-privileged modes.

Processor Modes

Mode	Privileged	Mode [4 3 2 1 0]
Abort (abt)	Yes	1 0 1 1 1
Fast Interrupt Request (fiq)	Yes	1 0 0 0 1
Interrupt Request (irq)	Yes	1 0 0 1 0
Supervisor (svc)	Yes	1 0 0 1 1
System (sys)	Yes	1 1 1 1 1
Undefined (und)	Yes	1 1 0 1 1
User (usr)	No	1 0 0 0 0

Register Access in Various Modes

innovate achieve lead

User and system					
R0 R1 . . R12 R13sp R14lr R15pc	Fast Interrupt request R8-fiq . . R14-fiq	Interrupt request R13_irq R14_irq	Supervisor R13_svc R14_svc	Undefined R13_undef R14_undef	Abort R13_abt R14_abt
Cpsr -	Spsr-fiq	Spsr_irq	Spsr_svc	Spsr_undef	Spsr_abt

Change of Mode

The following events can cause a mode change.

- Reset
- Interrupt Request
- Fast Interrupt Request
- Software Interrupt
- Data Abort
- Prefetch Abort
- Undefined instruction

Exceptions and Interrupts

- Exception and interrupts cause suspension of normal execution of sequential instructions and the program counter moves to a specific location.
- For example when an interrupt occurs a mode change occurs.
- Mode changes from user mode to interrupt request mode.
- r13 and r14 user registers are replaced by r13_irq and r14_irq.
- r14_irq contains the return address and r13_irq contains the stack pointer for interrupt request mode. Also saved program status register **spsr** takes the place of **cpsr**.
- **cpsr** contents get copied to **spsr_irq**

Instruction Sets

ARM : J and T bits in **cpsr** are 0 and the ARM core enters this state by default. ARM mode instructions are 32-bits wide. There are 58 core instructions. Most instructions are conditional instructions. Barrel shifter and ALU are used in this mode. Read-write in privileged mode is allowed. There are 15 general purpose registers that can be used in this mode.

Thumb: When T bit is 1, the Arm processor core enters the Thumb state In this mode, the instructions are 16-bits wide. There are 30 core instructions. Branch instructions are conditional in this mode. Separate barrel-shifter and ALU are used in this mode. There is no direct access to program status register. There are 8 general purpose registers, 7 high registers and PC in this mode.

Jazelle: In this mode, 8-bit Javacodes can be used. T=0, J=1 in the **cpsr** bits The instruction size is 8-bits. About 60% of the Java bytecodes are implemented in hardware, the rest in software.

Condition Flags

Flag	Name	Set under this condition
Q	Saturation	Result causes an overflow and/or saturation
V	Overflow	Result causes signed overflow
C	Carry	Unsigned carry in the result
Z	Zero	Result is zero
N	Negative	Bit 31 is 1

Conditional Execution Used by Instructions

Mnemonic	Name	Flag
EQ	Equal	Z
NE	Not equal	Z
CS/ HS	Carry Set / Unsigned higher or same	C
CC /LO	Carry clear/ Unsigned lower	c
MI	Minus/ Negative	N
PL	Plus/ Positive or zero	n
VS	Overflow	V
HI	Unsigned higher	z C
LS	Unsigned lower or same	Z or c
GE	Signed greater than or equal	NV or nv
LT	Signed less than	Nv or nV
GT	Signed greater than	NzV or nzv
LE	Signed less than or equal	Z or Nv or nV
AL	Always(unconditional)	ignored

Pipelining

The processor performs the following three operations on an instruction located in the program memory.

Fetch – The instruction is loaded from memory

Decode – The various fields in the instructions are identified and associated with specific operations.

Execute – The instructions are processed based on the decoded fields and the result may be returned or saved in a register.

- Only one of the operations may be performed on a single instruction.
- More than one instruction may participate in this process.
- One instruction may undergo execution while another instruction may be in the process of getting fetched from memory.
- The mechanism of speeding up the average execution process of an instruction starting from fetch operation to execution with simultaneous instructions being processed is called pipelining.

ARM Exceptions

innovate achieve lead

Exception Name	Vector Number	Priority
Reset	1	-3
NMI	2	-2
Hard Fault	3	-1
Reserved	4 to 10	Reserved
SVCall	11	Configurable
Reserved	12 and 13	Reserved
PendSV	14	Configurable
SysTick	15	Configurable
Interrupt (IRQ0-IRQ31)	16 to 47	Configurable

Exception Handling

- An exception is a condition that halts the sequential execution of instructions in the ARM core.
- The ARM core has a software exception handler that is invoked when an exception occurs.
- This specific software routine determines the cause of the exception and services the exception.
- The ARM core sets the program counter to a specific memory address that is part of the vector table.
- Software can set four levels of priorities on some of these exceptions and interrupts.
- The highest user configurable priority is 0 and the lowest priority is 3. The configurable priority of 0 is the fourth priority with -3 being the highest priority reserved for Reset mechanism. NMI has the priority of -2 and Hard fault has a priority of -1.

Exceptions

- The Reset exception is used to run the software for initializing the embedded system.
- When the ARM core encounters an instruction that it is unable to decode, the program counter is set to address 0x00000004.
- Execution of a SWI instruction causes the program counter to be set to 0x00000008 which is the vector location for the software interrupt.
- The ARM core allows a variety of peripherals to interrupt the normal course of instruction flow and make the core execute a subroutine corresponding to the interrupt source. The program counter then branches to the interrupt location 0x00000018 as shown in the table. The vector table starts at 0x00000000 or 0xFFFF0000 in some processors.

Instructions – a few samples

ADC Add with Carry		
Syntax	ADC{cond}{S} Rd, Rn, Op2	
Description	Add Rn and Op2 and Carry flag and store result to Rd.	
Condition Flags	If S is specified update flags: N, Z, C,V.	
Example	ADC R0,R1,R2; ADDS R0,R1,R2	Add the contents of R1 and R2 and store the result in R0. Add the contents of R1 and R2 and store the result in R0. Set the flags according to the result of the operation.

Instructions..

AND Logical AND operation.		
Syntax	AND{cond}{S} Rd, Rn, Op2	
Description	Load Rd with logical AND of Rn with Op2.	
Condition Flags	If S is specified, N, Z flags are updated.	
Example	AND R5,R6,#0987	Logically AND the contents of R6 with the number 0x987 and store the result in R5
	ANDS R5,R6,#0987	Logically AND the contents of R6 with the number 0x987 and store the result in R5. Update the flags N and Z

Assembly Language Program

innovate achieve lead

```
;Prasm1.s
;Add the contents of 8-bit data present in two registers R0,R1 from ;
    and store the result in R2

        MOVS R0,#30
        MOVS R1,#56
        ADD R1,R1,R0
        MOVS R2,R1
STOP    B STOP
```