INTRODUCTION TO 16/32 BIT PROCESSORS

FEATURES

- 32 bit processors execute 32 bit operation at a time in single cycle
- 32 bit processor having a subset of instructions for 16 bit coding is called 16/32 bit processor
- These processors are required in sophisticated and precision applications
- Normally superscalar architecture is implemented.
  It means that multiple instructions can be executed in a single cycle using pipelining techniques
- In pipelining ,instruction fetch, instruction decode and instruction execution for different instructions is performed simultaneously in single cycle
- RISC architecture is implemented
- RISC has the following characteristics (features)
  - Fixed length instruction
  - Standard execution time for instructions (single cycle)
  - Fewer number of instructions
  - Fewer number of addressing modes and instruction formats
  - Large register set
  - Hardware implementation
  - Easier implementation of pipeline

ARM Architecture and Organization

- ARM stands for Advanced RISC Machines
- ARM designed a family of RISC super scalar processor architectures based on VLSI
- ARM has various architectures like ARM2, ARM3, ARM6, ARM8, ARM9
- It has many variants like V1, V 2, V 3, V 3M, V 4, V 4T, V 5 etc
- V3 onwards full 32 bit addressing for both data and code is implemented
- M variables have long multiply and multiply and accumulate facility
- T variants have thumb instruction set.
- E variants have enhanced DSP instructions

ARM / THUMB Programming Model

- Implements both Princeton architecture (ARM7) and Harvard Architecture (ARM9)
- Word alignment can be big endian or little endian
- 16, 32 bit general purpose register (r0 to r15)
- GPR can be used for data computation or as index register
- RALU operates operations through registers only
- CPSR is current program status register. It is in addition to 16 GPRs. It contains flags N,Z,C and V in bits 31, 30, 29 and 28

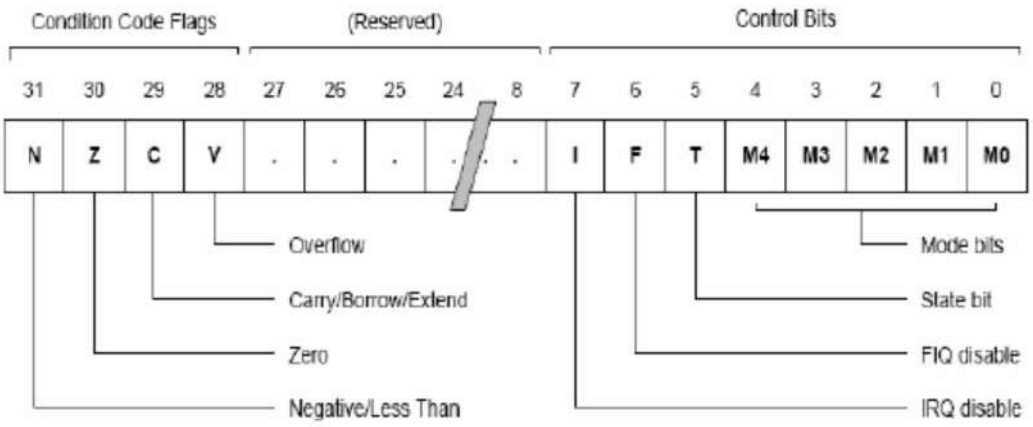
- R14 is used as return address pointer
- R15 is used as program counter
- R13 is used as implicit stack pointer
- Thumb programming model implements the following:
    - Eight registers r0 to r7 are used for most of the instructions
    - Memory is half word aligned
    - 16 bit instruction format is used
    - T bit of CPSR is used to switch from ARM to Thumb and vice versa
    - It does not have condition field (4 bits) in instruction format
    - It does not have shift/ rotate field (12 bits) in instruction format

ARM Architecture

1. RALU (Register based ALU): - The CPU of ARM processor is a three-address machine with two source operands and one destination operand. A source operand in RALU operations is first loaded, can also be first operated with rotate or shift instructions, can be either from register or register + memory or an 8-bit immediate operand. Other source operand from a register. After completion of the operation by RALU it sends the result to destination.

2. General purpose registers: - It has sixteen 32-bit registers. r13 is used as Stack Pointer, r14 contains return address called Link Register (LR), and r15 is used as Program Counter (PC).

3. Current program status register (CPSR): - It contains four status flags N (negative), Zero (Z), Carry (C) and overflow (V), and also it contains control and mode selection bits. The format of CPSR as follows



I = 1, disable Interrupt requests
F = 1, disable Fast interrupt
T = 0, Processor is in ARM state & T = 1, Processor is in Thumb state
N: set when the result is negative in two's complement arithmetic
Z: set when every bit of the result is zero
C: set when there is a carry out of the operation
V: set when an arithmetic operation results in an overflow

Mode (4:0): Processor Mode Operation Selection Bits

M[4:0]	Mode	Visible THUMB – State Registers	Visible ARM – State Registers
10000	User	r 0 – r 7, SP, LR, PC, CPSR	r 0 – r 14, PC, CPSR
10001	FIQ	r 0 – r 7, SP_FIQ, LR_FIQ, PC_FIQ, CPSR, SPSR_FIQ	r 0 – r 7, r 8_FIQ – r14_FIQ, CPSR, SPSR_FIQ
10010	IRQ	r 0 – r 7, SP_IRQ, LR_IRQ, PC_IRQ, CPSR, SPSR_IRQ	r 0 – r 12, r13_IRQ, r14_IRQ, PC, CPSR, SPSR_IRQ
10011	Supervisor	r 0 – r 7, SP_SVC, LR_SVC, PC_SVC, CPSR, SPSR_SVC	r 0 – r 12, r13_SVC, r14_SVC, PC, CPSR, SPSR_IRQ
10111	Abort	r 0 – r 7, SP_ABT, LR_ABT, PC_ABT, CPSR, SPSR_ABT	r 0 – r 12, r13_ABT, r14_ABT, PC, CPSR, SPSR_ABT
11011	Undefined	r 0 – r 7, SP_UND, LR_UND, PC_UND, CPSR, SPSR_UND	r 0 – r 12, r13_UND, r14_UND, PC, CPSR, SPSR_UND
11111	System	r 0 – r 7, SP, LR, PC, CPSR	r 0 – r 14, PC, CPSR

The remaining bits in CPSR are reserved, these bits values are not altered when the CPSR changing from one mode to other.

4. Saved program status register (SPSR): - It preserves the value of CPSR when exception occurs or call is made.

5. **Instruction Register (IR): -** It holds the current instruction opcode from four byte instruction queue.

6. **Instruction Decoder: -** Instruction decoder register to decode the IR opcode bits and activate the appropriate signals for hardwired implementation of instruction at IR.

7. Condition Test and Branch Logic: - It is an additional circuit to test the flags and initiate appropriate program flow path by branching.

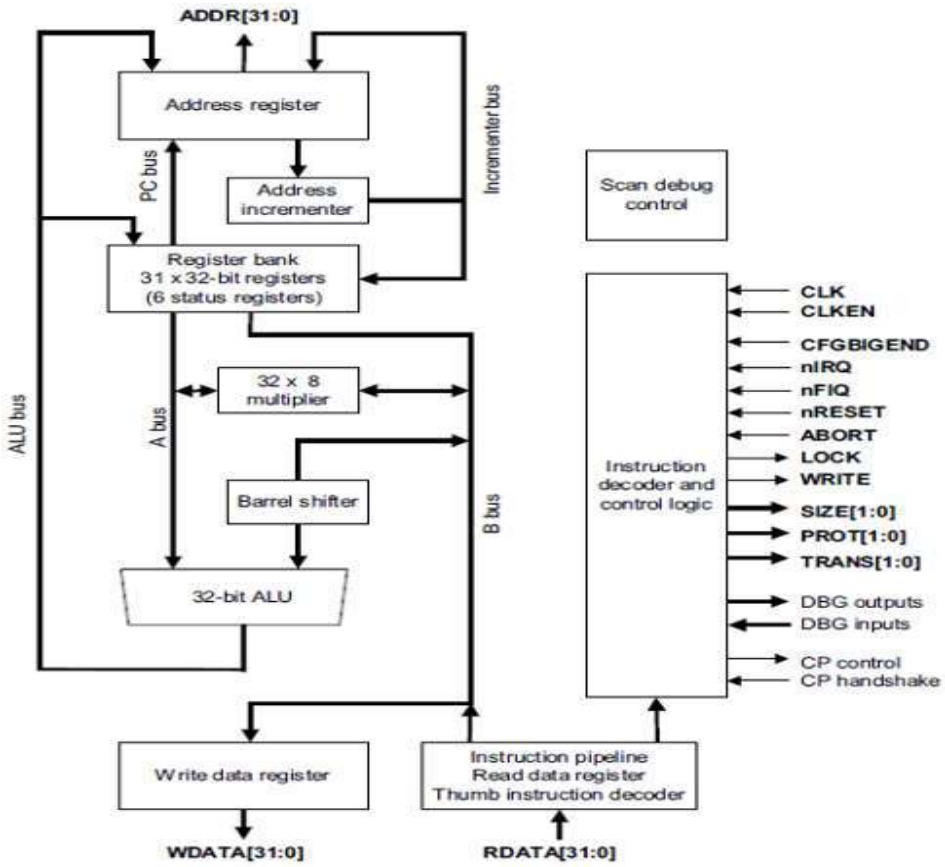
8. 32 X 32 multiply and 32 X 32 + 64 multiply and accumulate unit (MAC): – It improves performance for multiply, multiply and accumulate operation, necessary for DSP and control application

Features of ARM7 TDMI architecture

- It was designed by using HCMOS (High performance Complementary N – channel Metal Oxide Semiconductor) technology
- Its Die size of .25 Mm or less
- Low power consumption
- High performance 300 MIPS
- It has 32-bit data bus
- 32 bit address bus hence 4GB memory capacity
- It supports Princeton architecture

- It supports 8-bit / 16-bit / 32-bit data type
- It have 32 bit ARM instruction set and 16 bit thumb instruction subset
- It has thirty seven 32-bit registers
- It has 32-bit RALU
- It has high performance multipliers
- It has Co processor interface
- It has On chip JTAG interface
- It supports All RISC features
- It supports two types of interrupt requests
  1. Fast high priority interrupt request (FIQ) and
  2. Interrupt request (IRQ)

Architecture of ARM7TDMI Processor



32- Bit RALU: - ARM7TDMI have a 32-bit RALU (Register based ALU). By using this ALU the processor can perform all arithmetic and logical operations on register bank registers. There is no specific accumulator register for it. After completion of any operation by ALU result is send back to register bank registers. The result conditions are stored in to CPSR (Current Program Status Register).

Register bank Registers: - It has 37 registers; each register size is 32-bit. It consists of six Status registers (SPSRs) Saved Program Status Register. The remaining 31 registers forms register bank. Out of

31 register bank registers 16 registers are available for user in ARM state of processor, but in THUMB state of the processor only 8 registers are available.

Barrel Shifter: - It can perform shift or rotate operations on second source operand of RALU. It can perform shift or rotate operation any number of times in a single clock cycle. There is no separate shift and rotate operations related instructions in its either THUMB or ARM instruction sets.

32-bit Multiplier: - It has dedicated multiplier circuit by using this processor can perform multiply, multiply and accumulate related operations in a quick time**.**

It has two separate paths for receiving or transferring the information to or from other devices of the system by using data bus D [31:0].

The ARM7TDMI core uses a pipeline to increase the speed of the flow of instructions to the processor. This allows several operations to take place simultaneously, and the processing and memory systems to operate continuously.

A three – stage pipeline is used, so instructions are executed in three stages:

1. Fetch Stage 2. Decode Stage 3. Execute Stage

During normal operation, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The program counter points to the instruction being fetched rather than to the instruction being executed. This is important because it means that the Program Counter (PC) value used in an executing instruction is always two instructions ahead of the address.

Memory Access: - It has Von Neumann architecture, with a single 32-bit data bus carrying both instructions and data. Only load, store, and swap instructions can access data from memory. **The data can be: Byte (8-bits), Half Words (16-bit), and Words (32-bit).**

It is operates in two different states, those are **1. ARM state**

2. THUMB state.

It can be operate in six different modes. Their register organization of the processor in ARM state shown in the following figure.

ARM State General Registers and Program Counter

System & User	FIQ	Supervisor	About	IRG	Undefined
R0	R0	R0	R0	R0	R0
R1	R1	R1	R1	R1	R1
R2	R2	R2	R2	R2	R2
R3	R3	R3	R3	R3	R3
R4	R4	R4	R4	R4	R4
R5	R5	R5	R5	R5	R5
R6	R6	R6	R6	R6	R6
R7	R7	R7	R7	R7	R7
R8	R8_fiq	R8	R8	R8	R8
R9	R9_fiq	R9	R9	R9	R9
R10	R10_fiq	R10	R10	R10	R10
R11	R11_fiq	R11	R11	R11	R11
R12	R12_fiq	R12	R12	R12	R12
R13	R13_fiq	R13_svc	R13_abt	R13_irq	R13_und
R14	R14_fiq	R14_svc	R14_abt	R14_irq	R14_und
R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)	R15 (PC)

ARM State Program Status Register

CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

= *banked register*

It can be operate in six different modes. Their register organization of the processor in THUMB state shown in the following figure.

Exception handling

- Exceptions occur while a program is running. These are different from interrupts. Interrupts occur due to an execution of SW Instruction or due to request from hardware devices, internal or external.
- Exceptions occur due to run time conditions, For example detection of illegal opcode, and access to protected memory, division by zero, array out of bound or URL not found while connecting to Web
- Like interrupt vectors, exception vectors are available, as per CPU architecture, for servicing them. These vectors point to exception handling routines
- Exceptions have different priorities like interrupts
- The detection of exception and interrupting CPU is called catching exception
- Catching followed by vectoring to exception handler is called throwing the exception
- Processor status just before handling exception should be preserved
- Multiple exception can occur simultaneously
- ARM provides for interrupts and as well as exceptions

There are total of seven exceptions and interrupts given below:

	Type of Exception / Interrupt	High Vector Address	Use/Function
1.	**Reset**	FFFF0000H	CPU enters supervisory mode. Fast/ Normal interrupt disabled. PC is set to FFFF0000H or 00000000H if high vector not configured
2.	**Undefined Instruction**	FFFF0004H	CPU saves PC and CPSR. It disables normal Interrupts. PC is set to FFFF0004H. If co- processor instructions is encountered And there is no HW for this then SW evaluation is done on its occurrence
3.	**Software Interrupt (SWI)**	FFFF0008H	CPU enters supervisory mode. PC and CPSR saved. Normal interrupts disabled Particular function is executed on its occurrence
4.	**Pre-fetch Abort (Instruction fetch memory abort)**	FFFF000CH	CPU saves PC & CSPR, disables normal interrupts. Fetched instruction marked as invalid and pre fetch abort is flagged when this instruction reaches execution stage after fixing the reason for abort, return from handler.
5.	**Data Abort (Data Access memory abort)**	FFFF0010H	CPU saves PC and CSPR, disable normal Interrupts. Activating it in response to data access marks the data invalid. After fixing the reason return from handler. it occurs before any following inst has altered the CPU state
6.	**Interrupt request**	FFFF0018H	CPU saves PC and CPSR Generated externally by asserting IRQ pin at CPU Disables further IRQ's Executes ISR and returns IRQ's / FIQ's can be re enabled if needed
7.	**Fast Interrupt Request**	FFFF001CH	CPU saves PC and CPSR Disables FIQs and IRQs Generated externally by asserting FIQ pin at CPU FIQ has sufficient private registers hence no no need of saving register so context switch

overhead is minimized
FIQ handler can re enable IRQs / FIQs

Features of exception handling

- Handles exception after completely executing current instruction
- CPSR and PC saved to SPSR and R14
- To return, after handling exception, SPSR is moved to CPSR and R14 to PC through a suitable instruction. Eg. SUBS Pc, R14, #4
- Multiple exceptions allowed at same time
- Exception handler code should not cause further exception
- SWI interrupt handler can call another SWI interrupt handler

Exception / Interrupt Priorities

Exception/ Interrupt	**Priority**
Reset	1 Highest
Data Abort	2
FIQ	3
IRQ	4
Pre-fetch Abort	5
Illegal Inst / SWI	6 Lowest