**ARM Microcontroller**

**Introduction**

ARM belongs to the family of reduced instruction set computing architectures. ARM stands for advanced RISC machine. These processors have fewer transistors and has characteristics such as low power consumption, lesser cost and heat dissipation. ARM provides effective solution for lesser power consumption. ARM architecture supports 32-bit address space and arithmetic operations.

**Hardware block diagram**

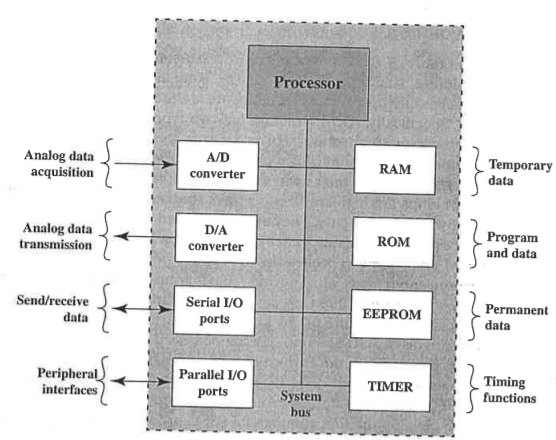


Figure 1: Microcontroller Chip Elements

The ARM processor has an A/D converter which converts analog signals to digital signals. The D/A converter converts analog to digital signal. There are serial and parallel I/O ports present. RAM stores temporary data. ROM is read only memory. EEPROM stands for electrically erasable programmable read only memory which is used to store lesser amounts of data. EEPROM need not be removed from the computer to be modified. Timer can be used for triggering an interrupt.

**Instruction Set**

ARM is a 32-bit architecture. ARM implements two instruction sets 32-bit ARM instruction set and 16-bit Thumb instruction set. The Thumb instruction set is a re-encoded subset of the ARM instruction set. Thumb is designed to increase the performance of ARM implementation that uses a 16-bit or narrower memory data bus. Most instructions are executed in a single step and can be conditionally executed. Up to 16 co-processors can be defined. Load/Store architecture is followed. There are three operand instruction formats.

**Operating Modes**

The ARM has six operating modes:

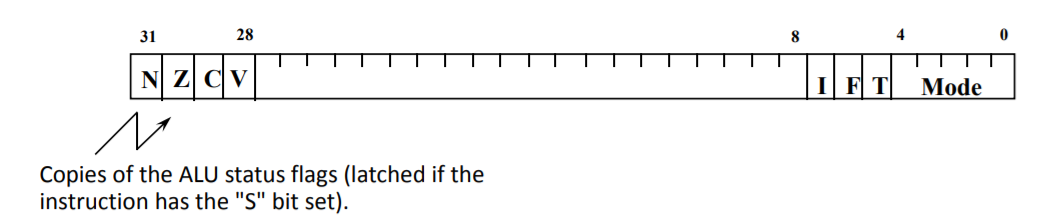
1. FIQ – Entered when highest priority interrupt is raised
2. IRQ – Entered when lowest priority interrupt is raised
3. Supervisor – Entered on reset and when software interrupt instruction is executed
4. Abort – to handle memory access violations
5. Undef – used to handle undefined instructions
6. User mode – is the usual ARM program execution state for executing normal programs
7. System – privileged mode using same registers as user mode

ARM has 37 registers in total, all of which are 32 bits long.

* 1 dedicated program counter
* 1 dedicated program status register
* 5 dedicated saved program status registers
* 30 general purpose registers

These registers are arranged into several banks with the accessible bank being governed by the processor mode. Each mode can access a set of 12 registers, r13 is the stack pointer and r14 is the link register. The program counter is r15 and current program status register. Privileged mode can also access saved program status register.

**Program status registers**



* Conditional code flags
* N = Negative result from ALU flag
* Z = Zero result from ALU flag
* C = Carried out flag
* V = overflow flag
* Interrupt disabled bits
* I = 1, disables the IRQ
* F= 1, disables the FIQ
* Mode bits – M[4:0] define the processor mode
* T bit (Architecture v4T only)
* T = 0, processor in ARM state
* T=1, processor in Thumb state

**Operation Types**

* Load and Store instructions – These instructions access memory locations. Arithmetic and logical instructions are performed only on registers and immediate values encoded in the registers. There are two types of instructions, load or store 32-bit word and 16bit unsigned halfword.
* Branch Instructions – These instructions allow conditional branch forwards or backwards up to 32MB. Branches are determined by 4-bit conditional field in the instruction.
* Data processing instructions – This category includes logical instructions for example: AND, OR, XOR and test and compare instructions.
* Multiply instructions – These instructions operate on word or half word operands and can produce normal or long results. For example: multiple instruction takes in two 32-bit operand and gives a 64-bit operand as a result.
* Parallel addition and subtraction instructions- Two operands are operated in parallel. These instructions are useful in image processing applications.
* Extend instructions – These are several instructions for unpacking data by sign zero extending bytes to halfwords or words and halfwords to words.
* Status register access instructions- ARM provides the ability and write the portions of the status register.

**ARM Addressing Modes**

ARM provides various variety of addressing modes. Load and store instructions are the only instructions that reference the memory. This referencing is done through base register plus the offset. There are alternatives to indexing:

* Load/Store Addressing

1. Offset – For this addressing mode indexing is not used. An offset is a value added to or subtracted from the base register to form the memory address. For example: STRB r0, [r1, #12]. In this example the offset is 12 which is the displacement. The resulting address is the location where the least significant byte from r0 is to be stored.
2. Preindex – The memory address is formed as the way as the offset addressing. For example: STRB r0, [r1, #12]!. The exclamation indicates pre-indexing.
3. Postindex- The memory address is the base register value. An offset is added or subtracted from the base register value and the result is written back to the base register. For example: STRB r0, [r1, #12].

* Data processing instruction addressing – use either register addressing or a mixture of register and instruction addressing. For register addressing value in the register operands may be scaled.
* Branch instructions – The only form of addressing for branch instruction is immediate addressing. The branch instruction contains a 24-bit value. For address calculation, this value is shifted by 2 bits so that the address is on a word boundary. The effective address range is 32GB from the program counter.
* Load/Store Multiple addressing – Load multiple instructions load a subset of the general-purpose registers from the memory. Store multiple instructions stores a subset of all general-purpose registers to memory. The list of registers for load or store is specified in a 16-bit field in the instruction with each bit corresponding to one of the 16 registers. Four addressing modes are used increment after, increment before, decrement after, decrement before.

**ARM Instruction format**

ARM architecture instructions are 32bit long. The first four bits of the instruction are condition codes. The next three bits specify general types of instruction. For most instruction other than the branch instruction, the next five consecutive bits is the opcode. The remaining 20 bits are used for operand addressing. To achieve a higher range of immediate values, the data processing immediate format specifies both immediate and rotate values.

THUMB instruction set is a reencoded subset of the ARM instruction set. Thumb instruction was designed to increase the speed of ARM instruction set. For example, by eliminating the use of conditional code field is not used. A 2-bit opcode field plus a 3-bit type field is used, thus saves 2 bits. The remaining 9 bits comes from operand specification.

Example of ARM instruction format

32 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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |

ror #0-range 0 through 0x000000FF-step 0x00000001

Data format of a thumb instruction set is shown below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Condition code | 001 | opcode | S | Rn | Rd | rotate | immediate |

**ARM Interrupt Handling Process**

1

1

M=USER32

I =0 , F=0

2

3

Return User Code

Software

Handler

M = IRQ , I=1 F=0

zklkl

zzM=IRQ

IRQ Interrupt

s

Figure 2: Interrupt Handling

Figure 2 shows state change when an IRQ occurs. The processor starts in state1 , which is in user mode. The IRQ bit (I-bit), within CPSR, is set to 0 allowing an IRQ to interrupt the processor. When an IRQ occurs the processor will automatically set the I-bit to 1 , masking any further IRQ. The F-bit remains set to 0, hence allowing the FIQ to interrupt the processor. FIQ are at a higher priority to IRQ and they should not be masked out. When the mode changes to IRQ mode the CPSR of the previous mode is automatically copied into IRQ SPSR. The software interrupt handler then takes over in state 3. When a state change in FIQ occurs, the processor goes through the same procedure as an IRQ interrupt but instead of just masking further IRQ(I-Bit) from occurring, the ARM processor also masks FIQ’s(F-bit). This means that both the I and F bits will be set to 1 when entering the software handler in state 3.

FIQ interrupt occurs when an external peripheral sets the FIQ pin to nFIQ. An FIQ interrupt is the highest priority interrupt. Upon entry into FIQ handler the core disables both IRQ’s and FIQ’s interrupts.

IRQ interrupt occurs when an external peripheral sets the IRQ pin. AN IRQ interrupt is the second highest priority interrupt. The IRQ handler will be entered if neither a FIQ interrupt or data abort event has occurred. Upon entry to the IRQ interrupts are disabled. The IRQ’s(I-bit) should remain disabled until the current interrupt source has been cleared.

Pre-fetch Abort event occurs when an attempt to load an instruction results in a memory fault. If an FIQ interrupt occurs, it can be taken while servicing the Pre-fetch abort.

SWI interrupt occurs when the SWI instruction has fetched and decoded successfully and none of the other higher priority exceptions/interrupts have been flagged. Upon entry to the handler the CPSR will be set to SVC mode.

**Controlling Interrupts**

**Enabling Interrupts**

The CPSR is first copied to r1, Then to enable IRQ interrupt bit 7 of the register is set to 0. The updated register is copied back to CPSR which enables the IRQ interrupts.

**Disabling Interrupts**

To disable IRQ interrupts bit 7 has to be set to 1 and similar procedures are followed as enabling the interrupts.

**Returning from an interrupt handler**

Due to processor pipeline, the return address from an interrupt or exception handler has to be manipulated. The address which is stored in the register will include an offset. This means that the value of the offset has to be subtracted from the link register.

|  |  |  |
| --- | --- | --- |
| Event | Offset | Return from Handler |
| Reset | N/A | n/a |
| Data Abort | -8 | SUBS pc,1r,#8 |
| FIQ | -4 | SUBS pc,1r,#4 |
| IRQ | -4 | SUBS pc,1r,#4 |
| Pre – fetch Abort | -4 | SUBS pc,1r,#4 |
| SWI | 0 | MOVS pc,1r |
| Undefined Instruction | 0 | MOVS pc,1r |

Interrupt Handling Steps:

1. External source sets the Interrupt flag processor masks further external interrupts and vectors to the interrupt handler via an entry in the vector table.
2. Upton the entry to the handler, the handler code Saves the current context of the non -banked registers
3. The handler then identifies the interrupt source and executes the appropriate interrupt service routine(ISR).
4. ISR services the interrupt
5. Upon return from the ISR the handler restores the context
6. Enables interrupts and return

**ARM Pipeline Processing**

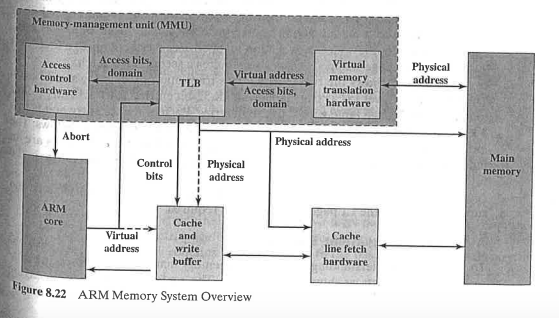
The ARM uses a pipeline to increase the speed of the processor. This allows several operations to take place simultaneously and the processing and the memory systems to operate continuously.

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

* Fetch
* Decode
* Execute

The instruction is fetched from the memory. This is followed by the second step, the registers used in the instruction are decoded followed by executing the instruction, to execute the instruction, the registers are read from the register bank, the shift ALU operations are performed and the registers are written back into to the register bank.

**ARM Memory Organization**



The above figure provides an overview of memory organization in ARM. The virtual memory translation hardware uses one or two levels of tables for translation from virtual to physical addresses. The translation lookaside buffer(TLB) is a cache of recent page table entries . If an entry is available in the TLB , then the TLB directly sends a physical address to main memory for a read or write operation. Entries in the translation also include access control bits which determine whether a given control hardware supplies an abort signal to the ARM processor.

**Memory Hierarchy**

The Memory Hierarchy is shown below:

* Registers
* On chip cache Memory
* 2nd level off chip cache
* Main Memory
* Hard disk

The memory capacity increases from Registers to hard disk. The Access time reduces from registers to hard disk. The cost increases from hard disk to registers.

**Memory Management**

When the processor generates a memory access, The Memory management unit (MMU)performs:

1. A lookup for the requested virtual address and current ASID and current security state in the relevant instruction or data TLB.
2. A hardware translation table walk if the lookup in step1 misses.

If the MMU finds a matching TLB entry, it uses the information in the entry as follows:

1. The access permission bits and the domain determine if the access is enabled. If the matching entry does not pass the permission checks, the MMU signals a memory abort.
2. The memory region attributes specified in the CP15 c10 registers control the cache and write buffer and determine if the access is secure or nonsecure, cached or noncached and device or shared
3. The MMU translates the virtual address to a physical address for the memory access.

If the MMU does not find a matching entry, a hardware table walk occurs.

**Applications of ARM Microcontroller**

Microcontrollers are used for various applications like power tools, implantable medical devices , automobile engine control systems, office machines, remote controls applications, toys and many more embedded systems. By dipping the size and expenditure in comparison to a design that make use of a different micro-processor, I/O devices and memory. Mixed signal micro-controllers are general putting together analog constitutes required controlling non digital electronic signals. Few of the applications are:

1. Light sensing and controlling devices
2. Temperature sensing and controlling devices
3. Fire detection and safety devices
4. Industrial instrumentation devices
5. Process control devices
6. Industrial instrumental devices
7. Process control devices
8. Volt Meter
9. Measuring revolving objects
10. Current meter
11. Hand held metering systems
12. Image processing
13. Patient Monitoring
14. Automotive power train
15. Anti-lock braking