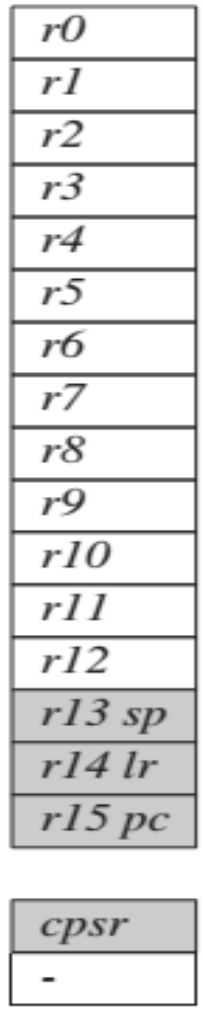
ARM ARCHITECTURE

REGISTERS

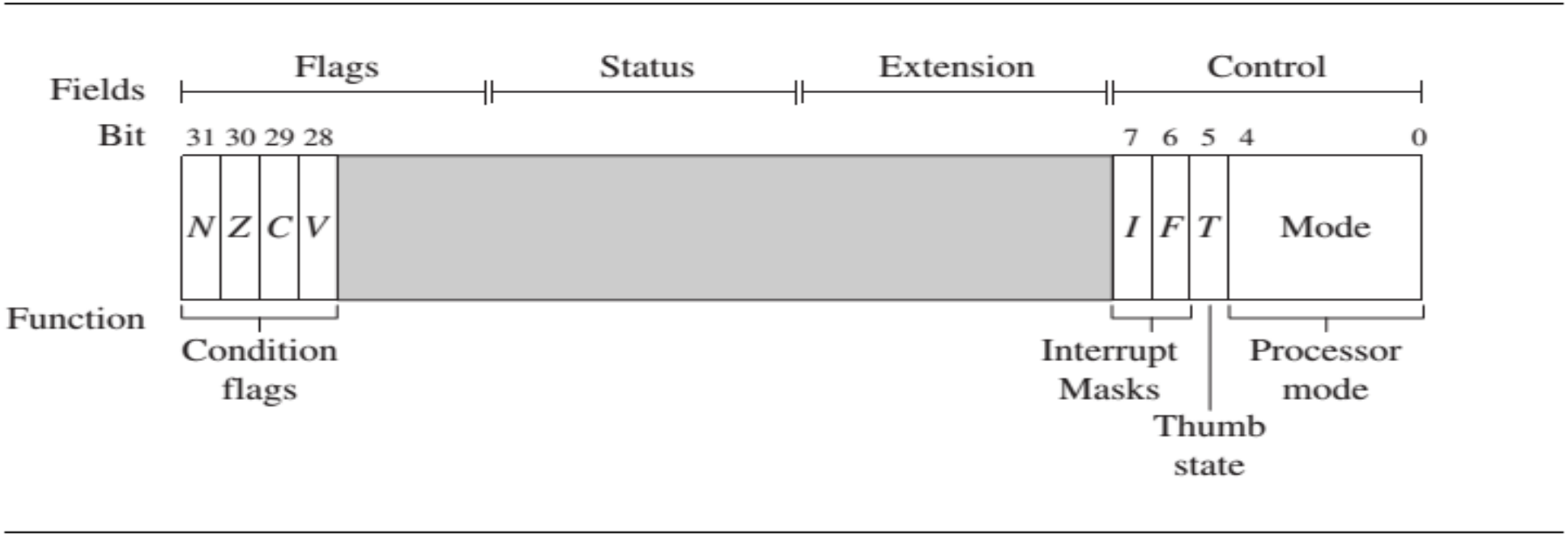
- General Purpose registers either hold address or data.
- Figure shows the registers available in user mode.
- All the registers are of 32 bit length.
- Register r13 is used as stack pointer.
- Register r14 is link register where core puts the return address whenever it calls a subroutine.
- Register r15 is the program counter and it contains the address of the next instruction to be fetched by the processor.



- Registers from r0 to r13 orthogonal.
- That is any instruction that is applicable to r0 is equally applicable to all other registers for r1 to r13.
- But there are instructions that treat registers r14 and r15 in a special way.

Current Program Status Register(CPSR)

- It is 32 bit register.

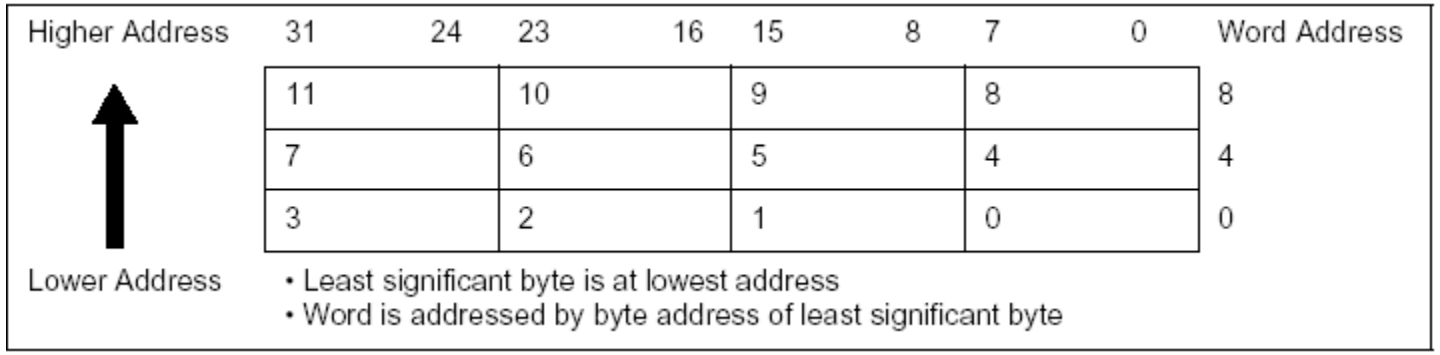


31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	*Z*	*C*	*V*	*Q*	*Res*		*J*	*Res*				*GE*[3:0]				*Res*						*E*	*A*	*I*	*F*	*T*	*mode*				

Field	Use
N	Negative flag, records bit 31 of the result of flag-setting operations.
Z	Zero flag, records if the result of a flag-setting operation is zero.
C	Carry flag, records unsigned overflow for addition, not-borrow for subtraction, and is also used by the shifting circuit. See Table A.3.
V	Overflow flag, records signed overflows for flag-setting operations.
Q	Saturation flag. Certain operations set this flag on saturation. See for example `QADD` in Appendix A (ARMv5E and above).
J	$J = 1$ indicates Java execution (must have $T = 0$). Use the `BXJ` instruction to change this bit (ARMv5J and above).
Res	These bits are reserved for future expansion. Software should preserve the values in these bits.
GE[3:0]	The SIMD greater-or-equal flags. See `SADD` in Appendix A (ARMv6).
E	Controls the data endianness. See `SETEND` in Appendix A (ARMv6).
A	$A = 1$ disables imprecise data aborts (ARMv6).
I	$I = 1$ disables IRQ interrupts.
F	$F = 1$ disables FIQ interrupts.
T	$T = 1$ indicates Thumb state. $T = 0$ indicates ARM state. Use the `BX` or `BLX` instructions to change this bit (ARMv4T and above).
mode	The current processor mode. See Table B.4.

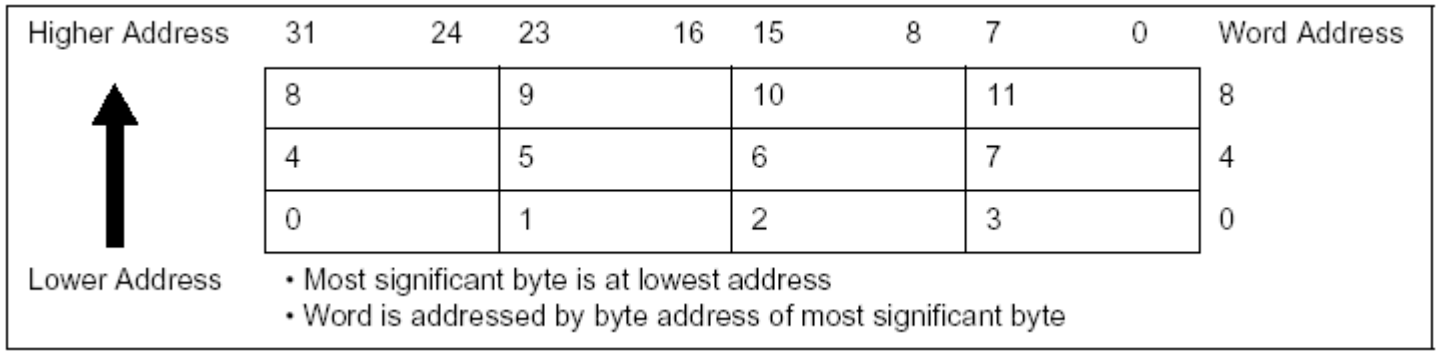
Little endian addresses of bytes within words

- In little endian format, the lowest numbered byte in a word is considered the word's least significant byte, and the highest numbered byte the most significant. Byte 0 of the memory system is therefore connected to data lines 7 through 0.



Big endian addresses of bytes within words

- In big endian format, the most significant byte of a word is stored at the lowest numbered byte and the least significant byte at the highest numbered byte. Byte 0 of the memory system is therefore connected to data lines 31 through 24.



Processor Modes

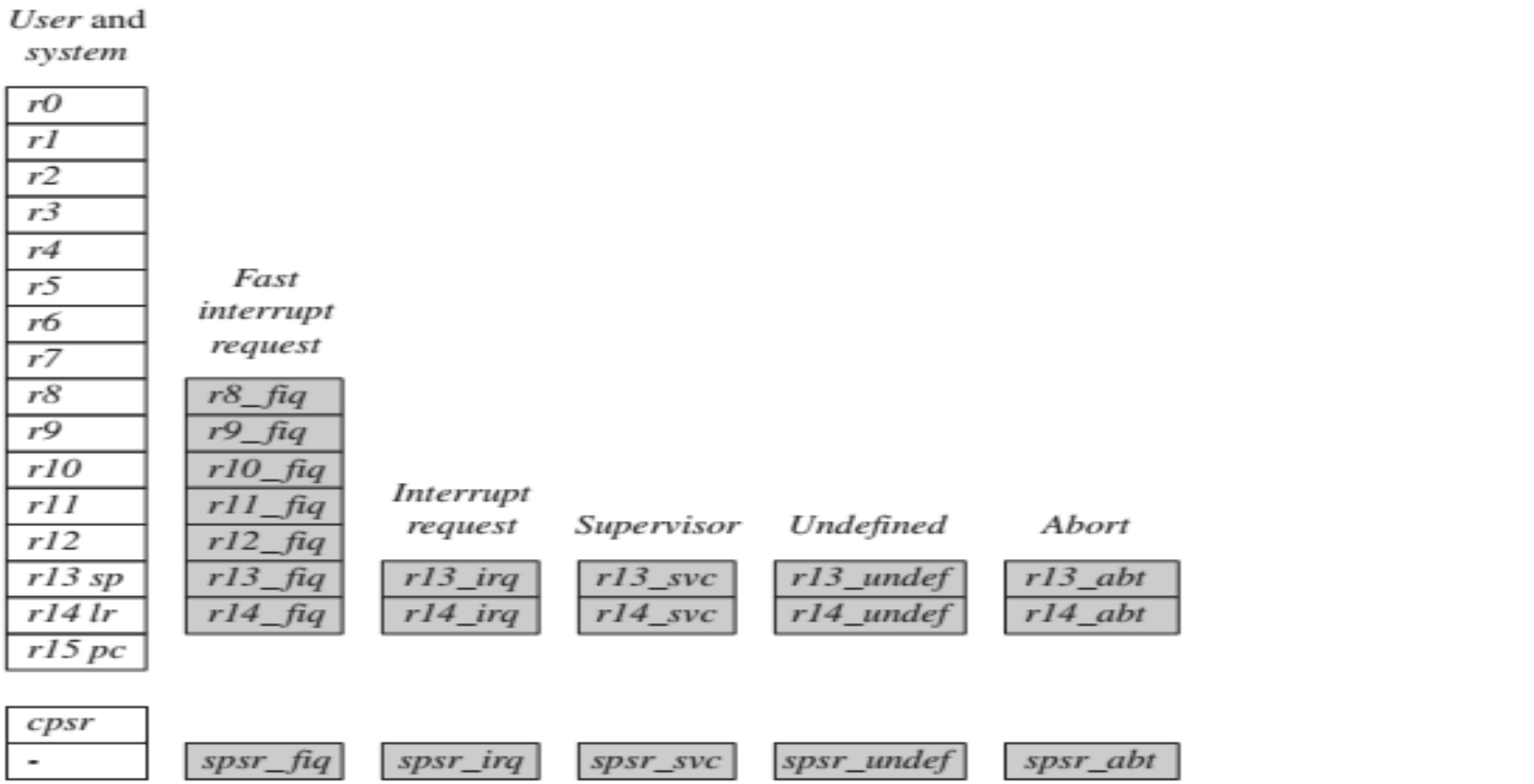
- Processor mode determines the access rights to the cpsr.
- Processor mode can be privileged or non privileged.
- A privileged mode allows full read write access to the cpsr.
- Non privileged mode allows only read access to the control field of the cpsr however it still allows the read write access to the conditional flags.
- There are total seven processor modes-six privileged and 1 non privileged.
- Abort ,fast interrupt request, interrupt request , supervisor, system and undefined are privileged modes.
- User is non privileged mode.

- The processor enters the abort mode when there is failed attempt to access memory.
- Fast interrupt request and interrupt request modes correspond to the two interrupt levels available on the ARM processor.
- Supervisor mode is the mode that the processor is in after reset and is generally the mode that an operating system kernel operates in.
- System mode is a special version of user mode that allows full read-write access to the cpsr.
- Undefined mode is used when the processor encounters an instruction that is undefined or not supported by the implementation.
- User mode is used for programs and applications.

Processor mode.

Mode	Abbreviation	Privileged	Mode[4:0]
Abort	abt	yes	10111
Fast interrupt request	fiq	yes	10001
Interrupt request	irq	yes	10010
Supervisor	svc	yes	10011
System	sys	yes	11111
Undefined	und	yes	11011
User	usr	no	10000

Banked Registers



- There are 37 registers in total.
- Out of which 20 are hidden from program at different times.
- These registers are called banked registers.
- They are only available when the processor is in particular mode.
- Every processor mode except user mode can change the mode by writing directly into mode bits of cpsr.
- On mode change a banked register from the new mode will replace an existing register.
- When power is applied processor starts with supervisor mode.

State and Instruction Set

- The state of the core determines which instruction set is being executed.
- There are three Instruction Sets.
- ARM, THUMB and Jazelle.
- The ARM instruction set is active only when the processor is in ARM State.
- Similarly the Thumb instruction set is only active when the processor is in Thumb state.
- You cannot intermingle sequential ARM, Thumb, and Jazelle instructions.
- The Jazelle J and Thumb T bits in cpsr reflect the state of the processor.
- Jazelle executes the 8 bit instructions and is a hybrid mix of software and hardware designed to speed up the execution of Java bytecoes.

ARM and Thumb instruction set features

	ARM (*cpsr* $T = 0$)	Thumb (*cpsr* $T = 1$)
Instruction size	32-bit	16-bit
Core instructions	58	30
Conditional execution[a]	most	only branch instructions
Data processing instructions	access to barrel shifter and ALU	separate barrel shifter and ALU instructions
Program status register	read-write in privileged mode	no direct access
Register usage	15 general-purpose registers +*pc*	8 general-purpose registers +7 high registers +*pc*

Interrupt Masks

- Interrupt Masks are used to stop specific interrupt requests from interrupting the processor.
- There are two interrupt request levels available on the ARM processor core.
- Interrupt request(IRQ) and fast interrupt request(FIQ).
- The cpsr has two interrupt mask bits, 7 and 6 (or I and F) which control the masking of IRQ and FIQ.
- The I bit masks IRQ when set to binary 1, and similarly the F bit masks FIQ when set to binary 1.

Conditional Flags

- Conditional Flags are updated by comparisons and the result of ALU operations that specify the S instruction suffix.
- For example. If a SUBS subtract instruction results in a register value of zero, the Z flag in the cpsr is set.
- Q flag is used for enhanced DSP instructions and it indicates the saturation or overflow.

Flag	Flag name	Set when
Q	Saturation	the result causes an overflow and/or saturation
V	oVerflow	the result causes a signed overflow
C	Carry	the result causes an unsigned carry
Z	Zero	the result is zero, frequently used to indicate equality
N	Negative	bit 31 of the result is a binary 1

Conditional Execution

- Most ARM instructions an be executed conditionally on the value of the condition flags.
- Conditional execution controls whether or not the core will execute an instruction.
- Most instructions have a conditional attribute that determines if the core will execute it based on the setting of the condition flags.
- Prior to execution the processor compares condition attributes with the condition flags in the cpsr.
- If they match then the instruction is executed otherwise the instruction is ignored.
- The condition attribute is post fixed to the instruction mnemonics which is encoded into the instruction.
- When a condition mnemonics is not present, the default behavior is to set it to always execute.

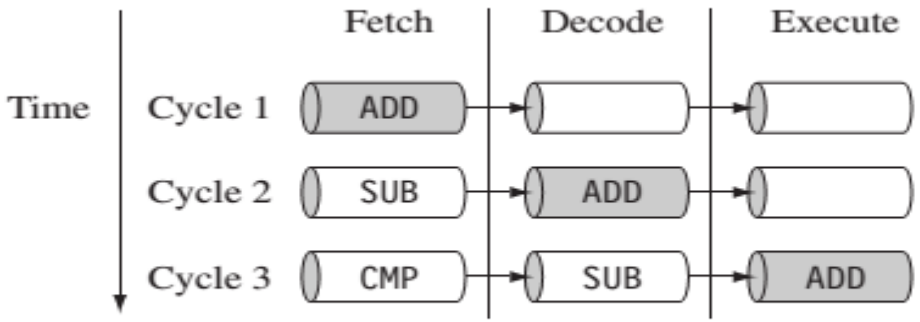
Condition mnemonics.

Mnemonic	Name	Condition flags
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*
VC	no overflow	*v*
HI	unsigned higher	*zC*
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

Pipeline

- A pipeline is the mechanism a RISC processor used to execute instructions.
- Using pipeline speeds up execution by fetching the next instruction while other instructions are being decoded and executed.
- Figure shows the ARM three stage pipeline
- Fetch loads an instruction from memory.
- Decode identifies the instruction to be executed.
- Execute processes the instruction and writes the result back to a register.

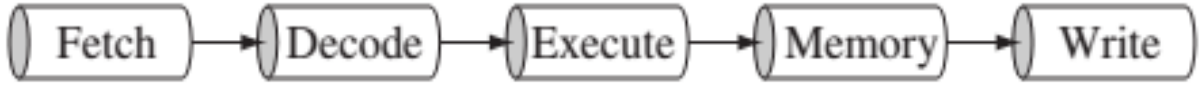
ARM7 Three-stage pipeline.



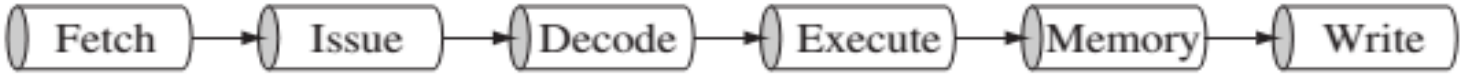
Pipelined instruction sequence.

- Figure above shows the pipeline using a simple example.
- It shows a sequence of three instructions being fetched, decoded and executed by the processor.
- Each instruction takes a single cycle to complete after the pipeline is filled.
- This is three stage pipe line procedure implemented in ARM 7.

- The pipeline design for each ARM family differs.
- ARM 9 uses 5 stage pipeline
- ARM 10 increases the pipeline length still further by adding a sixth stage.

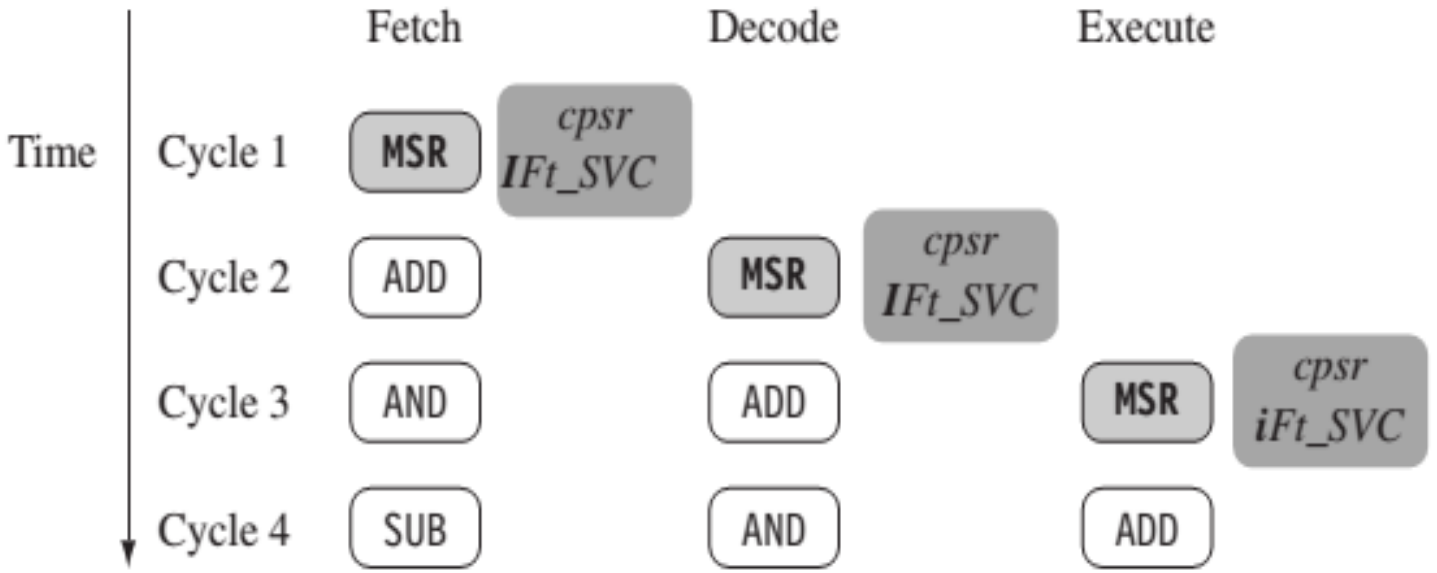


ARM9 five-stage pipeline.



ARM10 six-stage pipeline.

Pipeline Execution Characteristics



ARM instruction sequence.

Main characteristics of the pipeline

- The execution of a branch instruction or branching by the direct modification of the pc causes the ARM core to flush its pipeline.
- ARM 10 uses branch prediction which reduces the effect of a pipeline flush by predicting possible branches and loading the new branch address prior to the execution of the instruction.
- An instruction in the execute stage will complete even though an interrupt has been raised. Other instructions in the pipeline will be abandoned.

Exceptions, Interrupts and vector table

- When an exception or interrupt occurs, the processor sets the pc to a specific memory address.
- The address is within a special address range called the vector table.
- The entries in the vector table are the instructions that branch to specific routines designed to handle a particular exception or interrupt.

- The memory map address 0x00000000 is reserved for the vector table, a set of 32 bit words.
- On some processors the vector table can be optionally located at a higher address in memory.
- When an exception or interrupt occurs, the processor suspends normal execution and starts loading instructions from the exception vector table.
- Each vector table entry contains a form of branch instruction pointing to the start of a specific routine.

- Reset Vector is the location of the first instruction executed by the processor when power is applied. This instruction branches to the initialization code.
- Undefined instruction vector is used when the processor cannot decode an instruction.
- Software interrupt vector is called when you execute a SWI instruction. The SWI instruction is frequently used as the mechanism to invoke an operating system routine.
- Prefetch abort vector occurs when the processor attempts to fetch an instruction from an address without the correct access permissions. The actual abort occurs in the decode stages.

- Data abort vector is similar to prefetch abort but is raised when an instruction attempts to access data memory without the correct access permission
- Interrupt request vector is used by external hardware to interrupt the normal execution flow of the processor. It can only be raised if IRQ are not masked in the cpsr.
- Fast Interrupt vector is similar to the interrupt request but is reserved for hardware requiring faster response times. It can only be raised if FIQ are not masked in the cpsr.

The vector table.

Exception/interrupt	Shorthand	Address	High address
Reset	RESET	0x00000000	0xffff0000
Undefined instruction	UNDEF	0x00000004	0xffff0004
Software interrupt	SWI	0x00000008	0xffff0008
Prefetch abort	PABT	0x0000000c	0xffff000c
Data abort	DABT	0x00000010	0xffff0010
Reserved	—	0x00000014	0xffff0014
Interrupt request	IRQ	0x00000018	0xffff0018
Fast interrupt request	FIQ	0x0000001c	0xffff001c

Core Extensions

- There are standard components placed next to the ARM core.
- They improve performance, manage resources, and provide extra functionality.
- These are called core extensions.
- Cache and tightly coupled memory, memory management, and the coprocessor interface.
- The cache is a block of fast memory placed between main memory and the core.

- It allows for more efficient from some memory types.
- With a cache the processor core can run for the majority of the time without having to wait for data from slow external memory.
- A cache provides an overall increase in performance but at the expense of predictable execution.
- To predict the time taken by instructions another form of memory called tightly coupled memory(TCM).
- TCM is fast SRAM located close to the core and guarantees the clock cycles required to fetch instruction or data.

Memory Management

- Embedded systems often use multiple memory devices.
- Method is required to organize the devices and protect the system from applications trying to make inappropriate access to hardware.
- ARM cores have three different types of memory management hardware

  1. No extensions providing no protection
  2. A memory protection unit providing limited protection
  3. A memory management unit (MMU) providing full protection.

- Non protected memory is fixed and provide very little flexibility.
- It is normally used for small, simple embedded system that require no protection from rogue applications.

- MPUs employ a simple system that uses a limited number of memory regions.
- These regions are controlled with a set of special coprocessor registers, and each region is defined with specific access permissions.
- This type memory management is used for systems that require memory protection but don’t have a complex memory map.
- MMUs are the most comprehensive memory management hardware available on the ARM.
- The MMU uses a set of translation tables to provide fine grained control over memory.

Co Processor

- A coprocessor extends the processing features of a core by extending the instruction set or by providing configuration registers.
- More than one coprocessor can be added to the ARM core via the coprocessor interface.
- The coprocessor can also extend the instruction set by providing a specialized group of new instructions.
- These instruction are processed in the decode stage of the ARM pipeline.
- If the decode stage sees a coprocessor instruction it offers it to the relevant coprocessor.
- But if the coprocessor is not present or doesn’t recognize the instructions, then the ARM takes an undefined instruction exception.

Architecture Revisions

- Every ARM processor implementation executes a specific instruction set architecture (ISA).
- Code written to execute on an earlier architecture revisions will also execute on a later revision of the architecture.
- ARM processor nomenclature identifies individual processors and provides basic information about the feature set.

Nomenclature

- ARM uses the nomenclature to describe the processor implementations.
- The letters and numbers after the word “ARM” indicate the features a may have.
- Nomenclature does not include the. architecture revision information.

ARM{x}{y}{z}{T}{D}{M}{I}{E}{J}{F}{-S}

x—family
y—memory management/protection unit
z—cache
T—Thumb 16-bit decoder
D—JTAG debug
M—fast multiplier
I—EmbeddedICE macrocell
E—enhanced instructions (assumes TDMI)
J—Jazelle
F—vector floating-point unit
S—synthesizible version

ARM nomenclature.

Some more points about ARM Nomenclature

- All ARM cores after the ARM7TDMI include the TDMI features even though they may not include those letters after the “ARM” label.
- The processor family is a group of processor implementations that share the same hardware characteristics. For example, the ARM7TDMI, ARM740T, and ARM720T all share the same family characteristics and belong to ARM7 family.
- JTAG is described by IEEE 1149.1 Standard Test Access and boundary scan architecture. It is a serial protocol used by ARM to send and receive debug information between the processor core and test equipment.
- EmbeddedICE macrocell is the debug hardware built into the processor that allows breakpoint and watchpoints to be set.
- Synthesizable means that the processor core is supplied as source code that can be compiled into a form easily used by EDA tools.

Architecture Evolution

- The architecture has continued to evolve since the first ARM processor implementation was introduced in 1985.
- ARM has designed a number of processors that are grouped into different families according to the core they use.
- The families are based on the ARM7, ARM9, ARM10, and ARM11 cores.
- The postfix numbers 7, 9,10, and 11 indicate different core designs.
- The ascending number equates to an increase in performance and sophisticating.
- ARM 8 was developed but was soon superseded.

Revision	Example core implementation	ISA enhancement
ARMv1	ARM1	First ARM processor 26-bit addressing
ARMv2	ARM2	32-bit multiplier 32-bit coprocessor support
ARMv2a	ARM3	On-chip cache Atomic swap instruction Coprocessor 15 for cache management
ARMv3	ARM6 and ARM7DI	32-bit addressing Separate *cpsr* and *spsr* New modes—*undefined instruction* and *abort* MMU support—virtual memory
ARMv3M	ARM7M	Signed and unsigned long multiply instructions
ARMv4	StrongARM	Load-store instructions for signed and unsigned halfwords/bytes New mode—*system* Reserve SWI space for architecturally defined operations 26-bit addressing mode no longer supported
ARMv4T	ARM7TDMI and ARM9T	Thumb
ARMv5TE	ARM9E and ARM10E	Superset of the ARMv4T Extra instructions added for changing state between ARM and Thumb Enhanced multiply instructions Extra DSP-type instructions Faster multiply accumulate
ARMv5TEJ	ARM7EJ and ARM926EJ	Java acceleration
ARMv6	ARM11	Improved multiprocessor instructions Unaligned and mixed endian data handling New multimedia instructions

ARM7 Family

- ARM7 core has a Von Neumann style where both data and instructions use the same bus.
- The core has three stage pipeline and executes the architecture ARMv4T instruction set.
- ARM 7 is very popular core and it is used by many 32 bit ARM processors.
- It provides very good performance to power ratio.
- It is the first core to include the Thumb instruction set, a fast multiply instruction and EmbeddedICE technology.

CPU core	MMU/MPU	Cache	Jazelle	Thumb	ISA	E[a]
ARM7TDMI	none	none	no	yes	v4T	no
ARM7EJ-S	none	none	yes	yes	v5TEJ	yes
ARM720T	MMU	unified—8K cache	no	yes	v4T	no
ARM920T	MMU	separate—16K /16K $D + I$ cache	no	yes	v4T	no
ARM922T	MMU	separate—8K/8K $D + I$ cache	no	yes	v4T	no
ARM926EJ-S	MMU	separate—cache and TCMs configurable	yes	yes	v5TEJ	yes
ARM940T	MPU	separate—4K/4K $D + I$ cache	no	yes	v4T	no
ARM946E-S	MPU	separate—cache and TCMs configurable	no	yes	v5TE	yes
ARM966E-S	none	separate—TCMs configurable	no	yes	v5TE	yes
ARM1020E	MMU	separate—32K/32K $D + I$ cache	no	yes	v5TE	yes
ARM1022E	MMU	separate—16K/16K $D + I$ cache	no	yes	v5TE	yes
ARM1026EJ-S	MMU and MPU	separate—cache and TCMs configurable	yes	yes	v5TE	yes
ARM1136J-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes
ARM1136JF-S	MMU	separate—cache and TCMs configurable	yes	yes	v6	yes

More on ARM 7 family

- ARM7TDMI-S has the same operating characteristics as a standard ARM7TDMI but is also synthesizable.
- ARM720T is the most flexible member of the ARM7 family because it includes an MMU.
- The presence of MMU means the ARM720T is capable of handling the Linux and Microsoft embedded platform operating systems.
- ARM7EJ-S is also synthesizable and it ahs five stage pipeline.

ARM 9 family

- It has five stage pipeline.
- The processor can run at higher clock frequencies as compared to ARM7.
- The memory system uses the Hayward architecture which separates the data and instruction buses.
- The first processor in the ARM9 family was the ARM920T, which includes a separate D+I cache and an MMU.
- This processor can be used by operating systems requiring virtual memory support.
- The another core in the ARM9 product line is ARM926EJ-S.
- It is designed for use in small portable java enabled devices such as 3G phones and personal digital assistants.

ARM 10 and ARM 11

- It extends the ARM9 pipeline to six stages.
- It also supports an optional vector floating point unit which adds a seventh stage to the ARM 10 pipeline.
- The VFP significantly increases floating point performance.
- ARM1020E was the first processor to use ARM 10 core.
- The ARM 1136J-S was designed for higher performance and power efficient applications.
- It incorporates an eight stage pipeline with separate load store and arithmetic pipeline.