Seminar on

Architectures and Design Methods for Embedded Systems

Seminar Topic

Architecture of ARM Processor Family

Presented by Sambhavi Parajuli

INFOTECH II semester, University of Stuttgart

Supervisor Dominik Lücke

03.07.2006

Overview

1. History of development of the ARM processor.
2. Features of the ARM processor architecture and oganisation of the ARM components.
3. The ARM pipelines, modes and structure of the ARM components.
4. Development of wide range of the ARM processor families.
5. Instructions supported by the ARM processors.

History of the ARM Processor

- Developed the first ARM Processor (Acorn RISC Machine) in 1985 at Acorn Computers Limited.
- Established a new company named Advanced RISC Machine Limited and developed ARM6.
- Continuation of the architecture enhancements from the original architecture.

Features of the ARM Processor

- Incorporate features of Berkeley RISC design

  -a large register file

  -a load/store architecture

  -uniform and fixed length instruction field

  -simple addressing mode

Source: www.wikipedia.com

- Other ARM architecture features

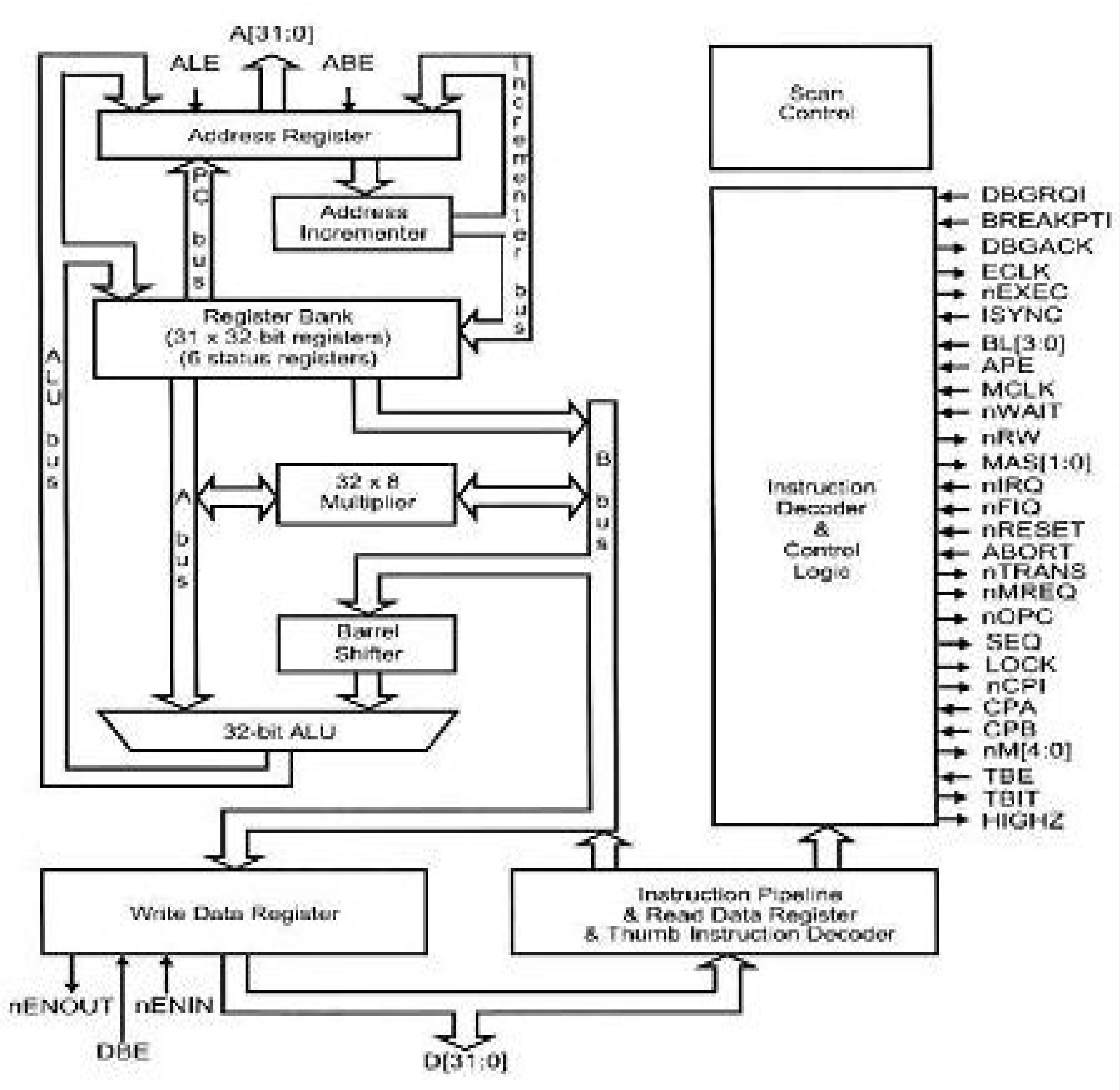
  -Arithmetic Logic Unit and barrel shifter

  -auto increment and decrement addressing mode

  -conditional execution of instructions

- Based on Von Neumaan Architecture or Harvard Architecture

Internal Organisation of the ARM Processor



Source: Shuqiang Zhang

ARM Pipelines

- Pipeline mechanism to increase execution speed.

- The pipeline design of each processor family is different.

ARM Processor Modes

ARM architecture supports seven operating modes: one unprivileged mode and six privileged modes.

Unprivileged mode

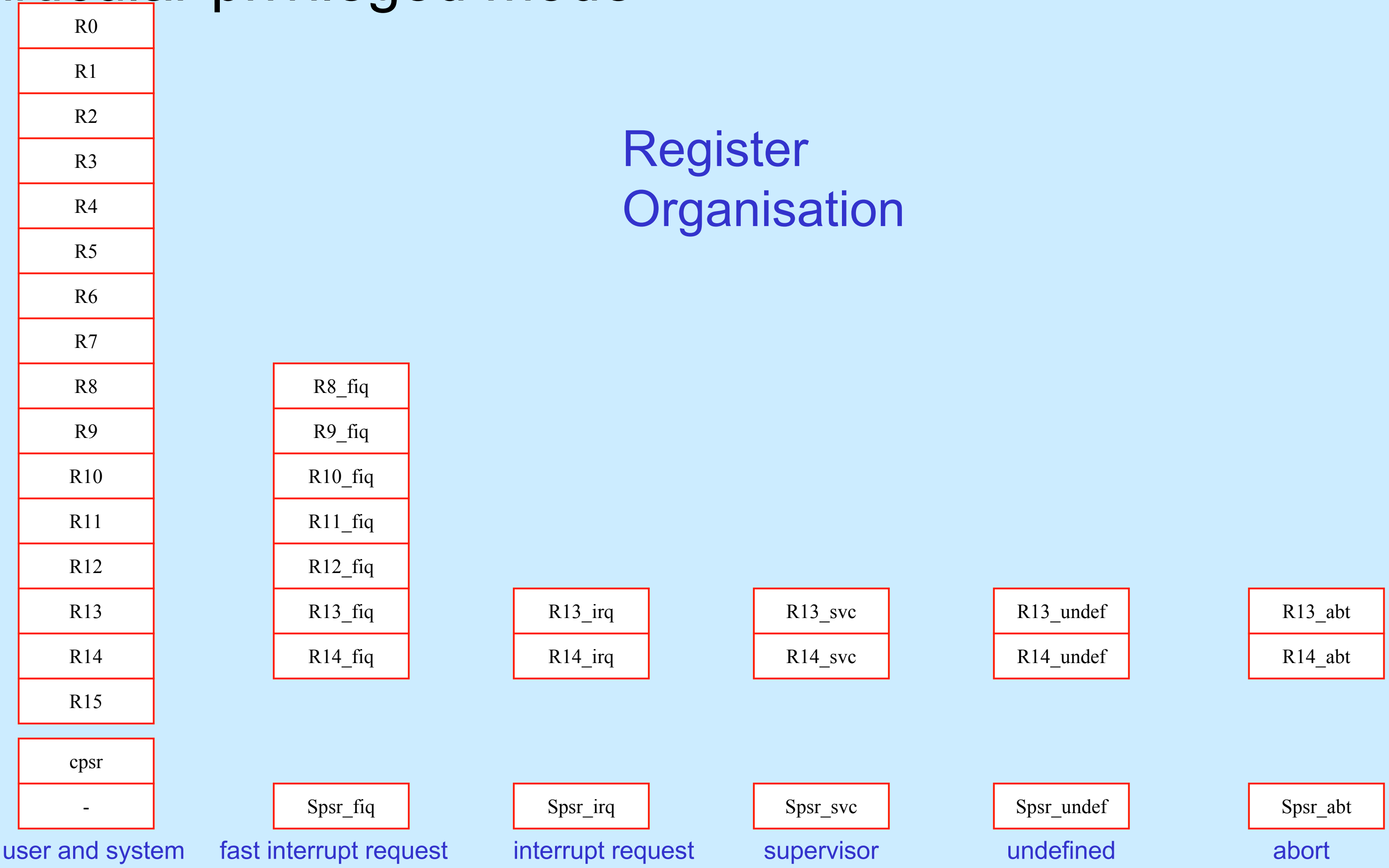
- User mode

Privileged mode

- Abort mode
- Fast Interrupt Request mode
- Interrupt Request mode
- Supervisor mode
- System mode
- Undefined mode

Register Files

- The ARM Registers-37 registers: one program counter, six program status registers, 30 general purpose registers.
- The Banked Registers: 20 registers when the processor is in a particular privileged mode

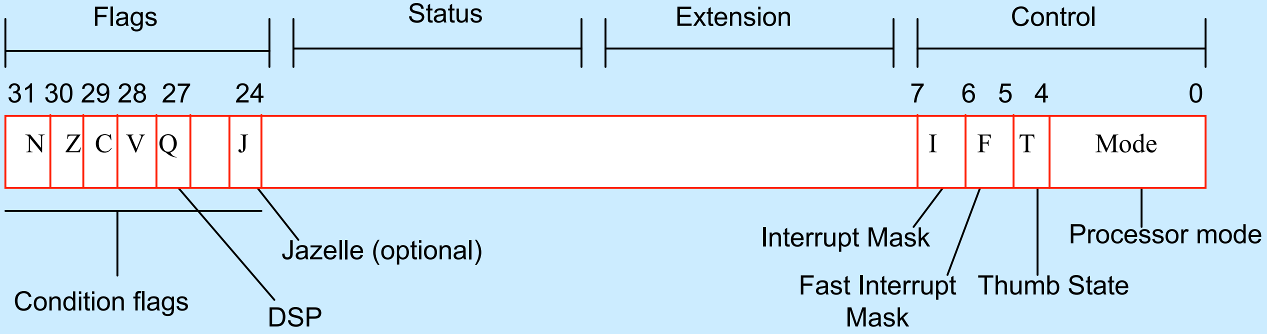


Register Organisation

user and system	fast interrupt request	interrupt request	supervisor	undefined	abort
R0					
R1					
R2					
R3					
R4					
R5					
R6					
R7					
R8	R8_fiq				
R9	R9_fiq				
R10	R10_fiq				
R11	R11_fiq				
R12	R12_fiq				
R13	R13_fiq	R13_irq	R13_svc	R13_undef	R13_abt
R14	R14_fiq	R14_irq	R14_svc	R14_undef	R14_abt
R15					
cpsr					
-	Spsr_fiq	Spsr_irq	Spsr_svc	Spsr_undef	Spsr_abt

Program Status Register

- Current Program Status Register



N=Negative, Z=Zero, C=Carry, V= oVerflow

Representation of Processor modes in cpsr

Processor Mode	Abbreviation	Mode[4:0]
Abort	abt	10111
Fast Interrupt Request	fiq	10001
Interrupt Request	irq	10010
Supervisor	svc	10011
System	sys	11111
Undefined	und	11011
User	usr	10000

- Saved Program Status Register

Instruction Set Design

ARM, Thumb and Jazelle

- ARM instruction set

  32-bit instructions,support load-store architecture,conditionally execution,use 3-address format

  Example : ADDS r0,r1,#1

- Thumb instruction set

  16-bit instructions, support load-store architecture, unconditionally execution (branch instructions), use 2-address format

  Example : ADD r1,#1

- Jazelle instruction set

  8-bit instructions, Javabyte code execution

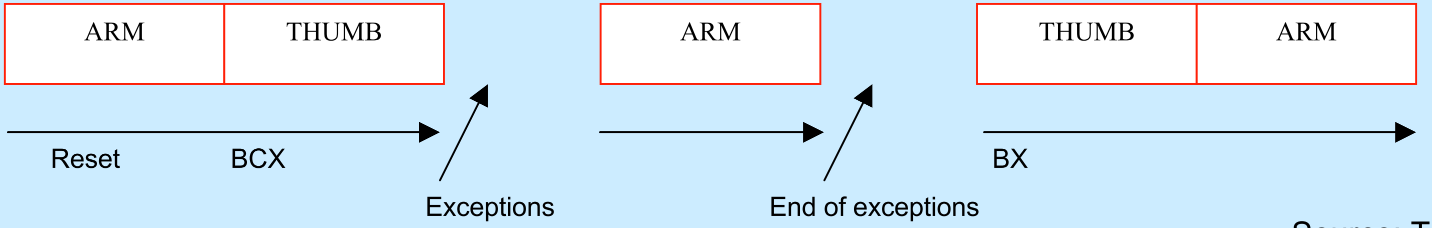
Interrupt and Exceptions

When an exception occurs, the processor stops normal execution and enters one of the exception modes.

Table1: The Interrupt Vector Table

Exception/Interrupt	Abbreviation	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

Exception when processor in the Thumb mode

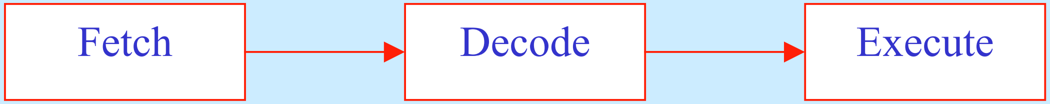


Source: Trevor Martin

The ARM Processor Families (I)

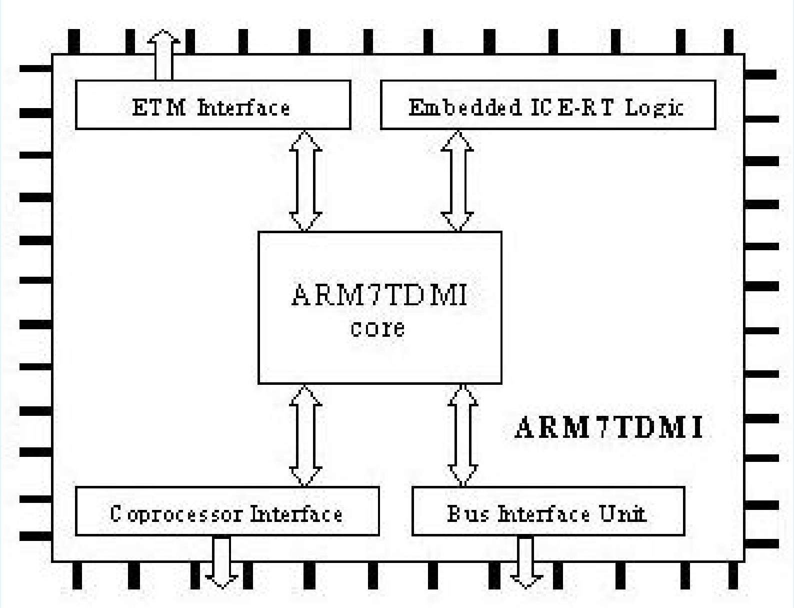
➢ The ARM7 Family

- 32-bit RISC Processor.
- Support three-stage pipeline

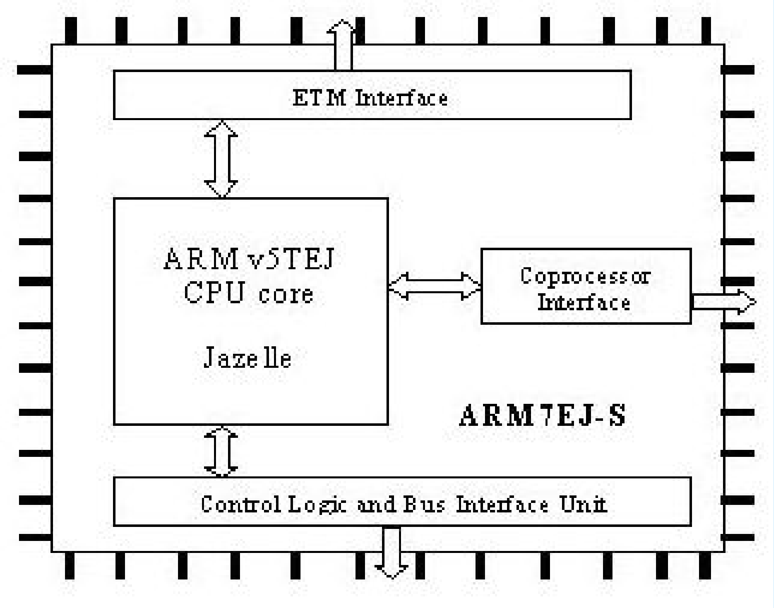


Fetch → Decode → Execute

- Uses Von Neumann Architecture.



ARM7TDMI Processor



ARM7EJ-S Processor

Source: www.arm.com

Characteristics of ARM7 family

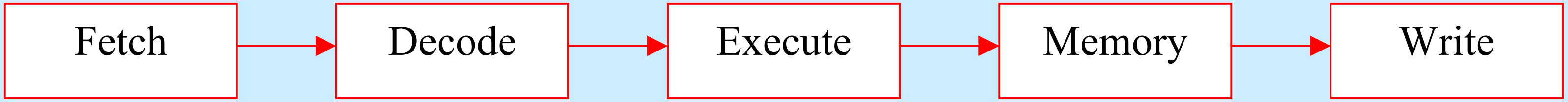
	Cache size(Inst/ Data	Tightly Coupled Memory	Memory Mgmt	Thumb	DSP	Jazelle
ARM720T	8k unified	-	MMU	Yes	No	No
ARM7EJ-S	-	-	-	Yes	Yes	Yes
ARM7TDMI	-	-	-	Yes	No	No
ARM7TDMI-S	-	-	-	Yes	No	No

Source: www.arm.com

Widely used in many applications such as palmtop computers, portable instruments, smart card.

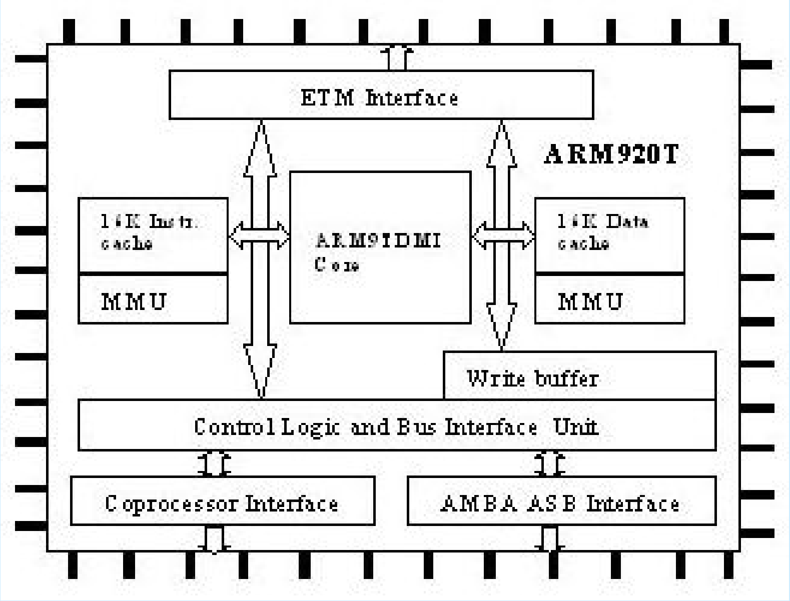
The ARM Processor Families (II)

- The ARM9 Family
- 32-bit RISC Processor with ARM and Thumb instruction sets
- Supports five-stage pipeline.

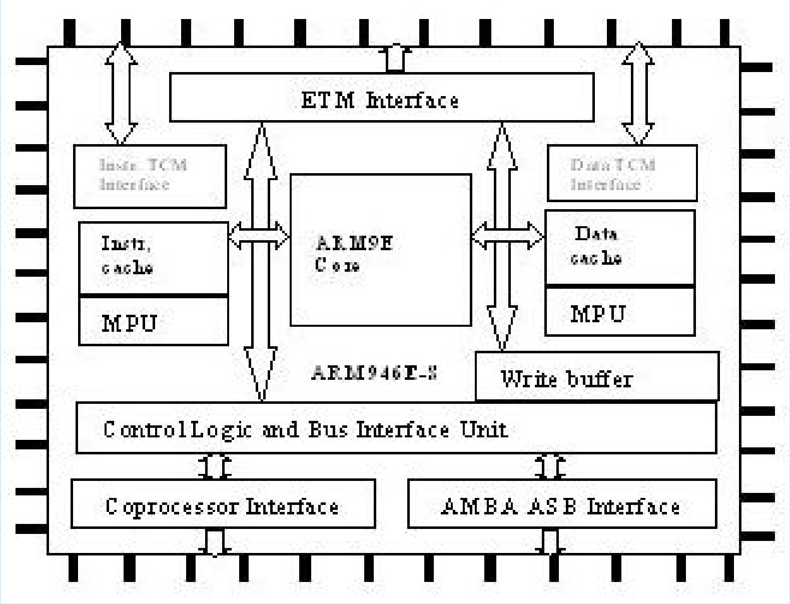


Fetch → Decode → Execute → Memory → Write

- Uses Harvard architecture



ARM920T Processor



ARM946E-S Processor

Source: www.arm.com

Characteristics of ARM9 Thumb Family

	Cache size(Inst/Data)	Tightly Coupled Memory	Memory Mgmt	Thumb	DSP	Jazelle
ARM920T	16k/16k	-	MMU	Yes	No	No
ARM922T	8k/8k		MMU	Yes	No	No

Characteristics of ARM9E Family

	Cache size(Inst/ Data)	Tightly Coupled Memory	Memory Mgmt	Thumb	DSP	Jazelle
ARM926EJ-S	Variable	Yes	MMU	Yes	Yes	Yes
ARM946E-S	Variable	Yes	MPU	Yes	Yes	No
ARM966E-S	-	Yes	-	Yes	Yes	No
ARM968E-S	N/a	Yes	DMA	Yes	Yes	No
ARM996H-S			MPU	Yes	Yes	No

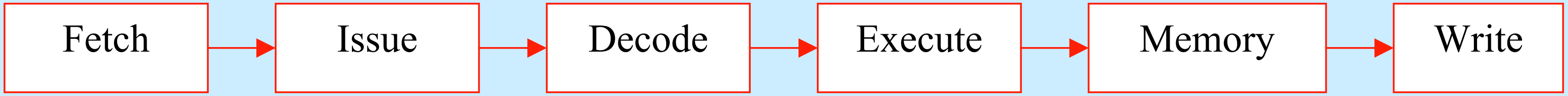
Source: www.arm.com

Widely used in mobile phones, PDAs,digital cameras, automotive systems, industrial control systems.

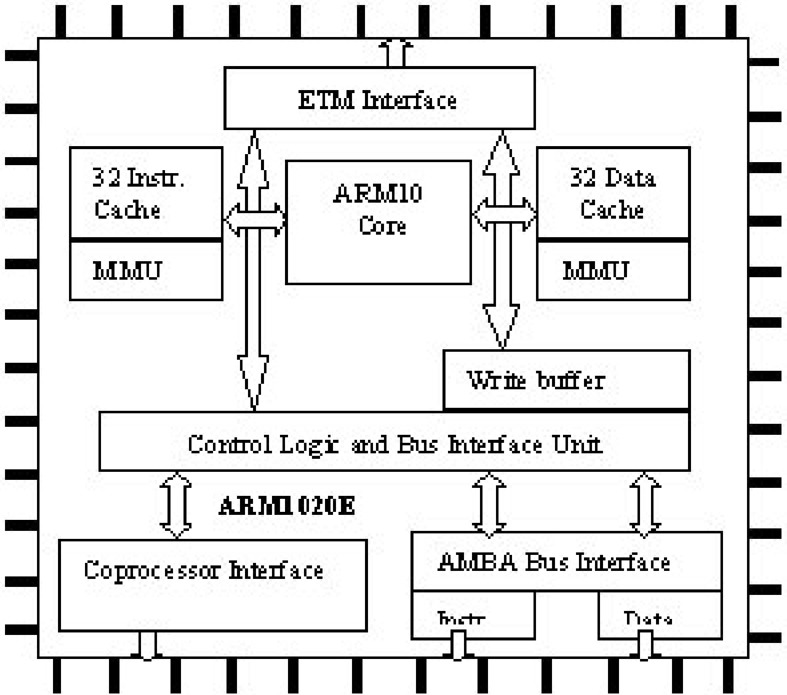
The ARM Processor Families (III)

The ARM10 Family

- 32-bit RISC processor with ARM, Thumb and DSP instruction sets.
- Supports six-stage Pipelines.



- Uses Harvard Architecture



ARM1020E Processor

Source: www.arm.com

Characteristics of ARM10 family

	Cache size(Inst /Data)	Tightly Coupled Memory	Memory Mgmt	Thumb	DSP	Jazelle
ARM1020E	32k/32k	-	MMU	Yes	Yes	No
ARM1022E	16k/16k	-	MMU	Yes	Yes	No
ARM1026EJ-S	Variable	Yes	MMU or MPU	Yes	Yes	Yes

Widely used in videophone, PDAs, set-top boxes, game console, digital video cameras,automotive and industrial control systems

The ARM Processor Families (IV)

➢ The ARM11 Family

- 32-bit RISC processor with ARM, Thumb and DSP instruction sets.
- Uses Harvard Architecture.
- Supports eight-stage Pipelines except ARM1156T2 uses nine-stage pipeline.
- Widely used in automotive and industrial control systems, 3D graphics, security critical applications.

Characteristics of ARM11 family

	Cache size(Inst /Data)	Tightly Coupled Memory	Memory Mgmt	Thumb	DSP	Jazelle
ARM11 MPCore	Variable	Yes	MMU+cache	Yes	Yes	Yes
ARM1136J(F)-S	Variable	Yes	MMU	Yes	Yes	Yes
ARM1156T2(F)-S	Variable	Yes	MPU	Yes	Yes	No
ARM1176JZ(F)-S	Varaible	Yes	MMU+TrustZone	Yes	Yes	Yes

Source: www.arm.com

Instructions supported by the ARM Processors

- ARM Instruction Set: standard 32-bit instruction set
- Thumb Instruction Set: 16-bit instruction set
- Jazelle Instruction Set: 8-bit instruction set

ARM Instruction Set supports six different types of instructions

- Data Processing Instructions
- Branch Instructions
- Load/Store Instructions
- Software Interrupt Instruction
- Program Status Register Instructions
- Coprocessor Instructions

Data Processing Instructions

used to manipulate data in general-purpose registers, employ a 3-address format, support barrel shifter.

Arithmetic Instructions:

ADD, ADC, SUB, SBC, RSB, RSC

Move Instructions:

MOV, MVN

Bit-Wise Logical Instructions:

AND, EOR, ORR, BIC

Comparison Instructions:

TST, TEQ, CMP, CMN

Multiply Instructions:

MUL, MLA

Examples of Data Processing Instructions

- ADD operation without barrel shifter

```
Pre     r0 = 0x00000000
        r1 = 0x00000002
        r2 = 0x00000001
ADD r0, r1, r2
Post    r0 = 0x00000003
```

- ADD operation with barrel shifter

```
Pre     r0 = 0x00000000
        r1 = 0x00000005
ADD r0, r1, r1, LSL #1
Post    r0 = 0x0000000f
        r1 =0x00000005
```

Barrel Shifter operations

Mnemonic	Description
LSL	Logical shift left
LSR	Logical shift right
ASR	Arithmetic right shift
ROR	Rotate right
RRX	Rotote right extended

Branch Instructions

Change the flow of sequencial execution of instructions and force to modify the program counter.

Branch (B)

jumps in a range of +/- 32 MB.

Branch with link(BL)

suitable for subroutine call by storing the address of next instructions after BL into the link register and restore the program counter from the link register while returning from subroutine.

Branch Exchange and Branch Exchange Link

for switching the processor state from Thumb to ARM and vice versa

ARM ↔ Thumb

Load/Store Instructions

Transfer data between memory and registers

Single Register Transfer Instructions

- used to move a single data item in and out of register (signed, unsigned, 16-bit half words and 32-bit word)
- supports register indirect, base-plus-offset and stack addressing mode
- LDR, STR, LDRB, STRB, LDRH, STRH, LDRSB

Multiple Register Transfer Instructions

- any subset or all the 16 registers loaded from or stored to memory but increase interrupt latency.
- addressing modes- IA, IB,DA, DB
- stack operations- FA, FD, EA, ED
- LDM, STM

Swap Instructions

- swap the contect of memory with the content of registers.
- SWP, SWPB

Software Interrupt Instruction

- Used to call the SWI exception handler (operating system functions).
- Forces processor into supervisor mode.
- SWI

Program Status Register Instructions

- Used to transfer the content of program status registers to/from a general-purpose register.
- MRS (copy program status register to a general purpose register), MSR(move a general-purpose register to a program status register)

Coprocessor Instructions

- Used to extend the instruction set, to control on-chips functions (caches and memory management) and for additional computations.
- CDP (data processing), MRC/MCR (register transfer), LDC/STC (memory transfer).

Systems with ARM Processor

Conclusion

- Continuous evolution of the ARM processors.
- Use of various design techniques such as RISC architectures, pipelines, DSP extension and Jazelle technology.
- High performance, lower power consumption and system cost, low silicon area and time-to-market.
- Provide benefits in the wide area of technology design and developments such as embedded real time applications, automotive control systems, portable applications and secure applications.

Thank you very much for your attention.

Any questions?