ARM

The first encounter

Authors:
Nemanja Perovic, nemanjaizbg@yahoo.com
Prof. Dr. Veljko Milutinovic, vm@etf.bg.ac.yu

What Is ARM?

- Advanced RISC Machine
- First RISC microprocessor for commercial use
- Market-leader for low-power and cost-sensitive embedded applications

ARM Powered Products

Features

- Architectural simplicity

which allows

- Very small implementations

which result in

- **Very low power consumption**

The History of ARM

- Developed at Acorn Computers Limited, of Cambridge, England, between 1983 and 1985
- Problems with CISC:
  - Slower then memory parts
  - Clock cycles per instruction

The History of ARM (2)

- Solution – the Berkeley RISC I:
  - Competitive
  - Easy to develop (less than a year)
  - Cheap
  - Pointing the way to the future

ARM Architecture

- Typical RISC architecture:
  - Large uniform register file
  - Load/store architecture
  - Simple addressing modes
  - Uniform and fixed-length instruction fields

ARM Architecture (2)

- Enhancements:
  - Each instruction controls the ALU and shifter
  - Auto-increment
    and auto-decrement addressing modes
  - Multiple Load/Store
  - Conditional execution

ARM Architecture (3)

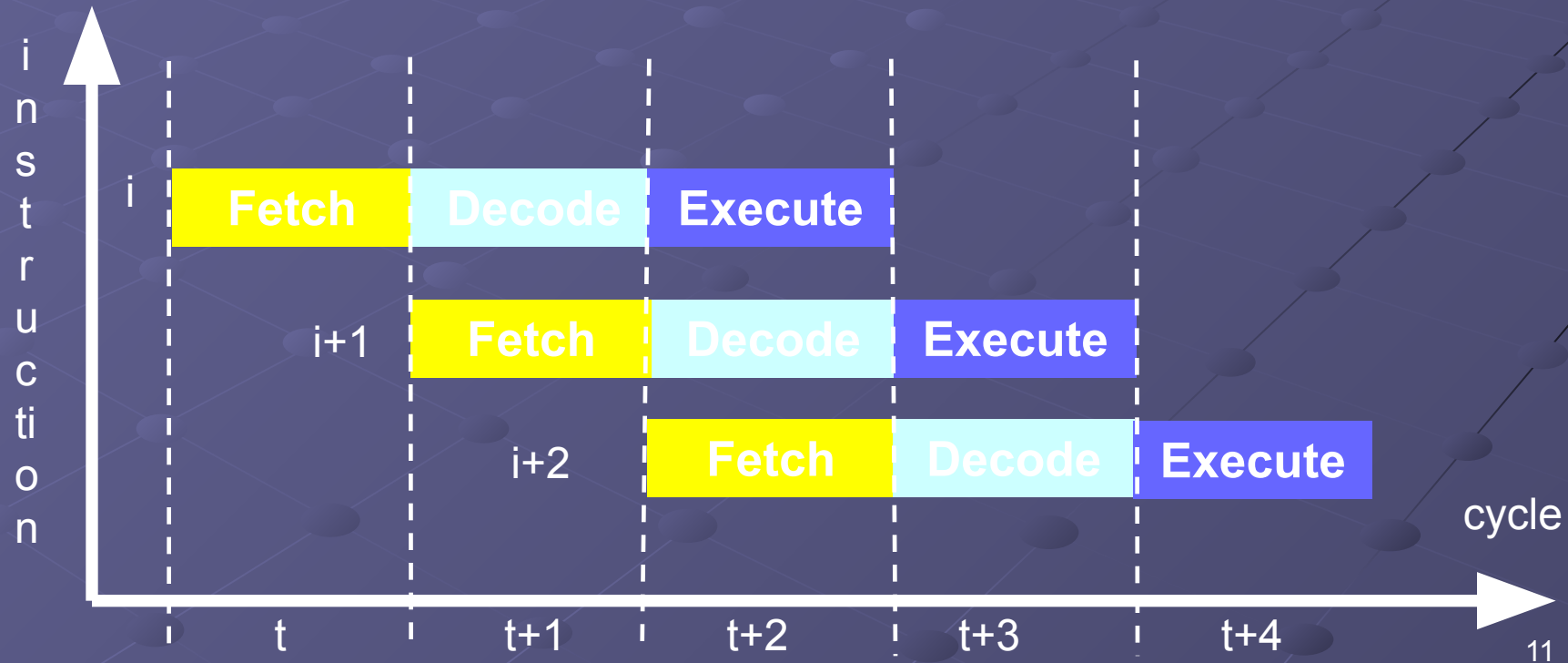
- Results:
  - High performance
  - Low code size
  - Low power consumption
  - Low silicon area

Pipeline Organization

- Increases speed –
  most instructions executed in single cycle
- Versions:
  - 3-stage (ARM7TDMI and earlier)
  - 5-stage (ARMS, ARM9TDMI)
  - 6-stage (ARM10TDMI)

Pipeline Organization (2)

- 3-stage pipeline: Fetch – Decode - Execute
- Three-cycle latency,
  one instruction per cycle throughput



Pipeline Organization (3)

- 5-stage pipeline:
  - Reduces work per cycle => allows higher clock frequency
    - Separates data and instruction memory => reduction of CPI (average number of clock Cycles Per Instruction)
- Stages:

Pipeline Organization (4)

- Pipeline flushed and refilled on branch, causing execution to slow down
- Special features in instruction set eliminate small jumps in code to obtain the best flow through pipeline

Operating Modes

- Seven operating modes:
  - User
  - Privileged:
    - System (version 4 and above)
    - FIQ
    - IRQ
    - Abort
    - Undefined
    - Supervisor

exception modes

Operating Modes (2)

User mode:

- Normal program execution mode
- System resources unavailable
- Mode changed by exception only

Exception modes:

- Entered upon exception
- Full access to system resources
- Mode changed freely

Exceptions

Exception	Mode	Priority	IV Address
Reset	Supervisor	1	0x00000000
Undefined instruction	Undefined	6	0x00000004
Software interrupt	Supervisor	6	0x00000008
Prefetch Abort	Abort	5	0x0000000C
Data Abort	Abort	2	0x00000010
Interrupt	IRQ	4	0x00000018
Fast interrupt	FIQ	3	0x0000001C

Table 1 - Exception types, sorted by Interrupt Vector addresses

ARM Registers

- 31 general-purpose 32-bit registers
- 16 visible, R0 – R15
- Others speed up the exception process

ARM Registers (2)

- Special roles:
  - Hardware
    - R14 – Link Register (LR): optionally holds return address for branch instructions
    - R15 – Program Counter (PC)
  - Software
    - R13 - Stack Pointer (SP)

ARM Registers (3)

- Current Program Status Register (CPSR)
- Saved Program Status Register (SPSR)
- On exception, entering *mod* mode:
  - (PC + 4) □ LR
  - CPSR □ SPSR_mod
  - PC □ IV address
  - R13, R14 replaced by R13_mod, R14_mod
  - In case of FIQ mode R7 – R12 also replaced

ARM Registers (4)

System & User	FIQ	Supervisor	Abort	IRQ	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_fiq	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)
CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
	SPSR_fiq	SPSR_svc	SPSR_abt	SPSR_irq	SPSR_und

Instruction Set

- Two instruction sets:
  - ARM
    - Standard 32-bit instruction set
  - THUMB
    - 16-bit compressed form
    - Code density better than most CISC
    - Dynamic decompression in pipeline

ARM Instruction Set

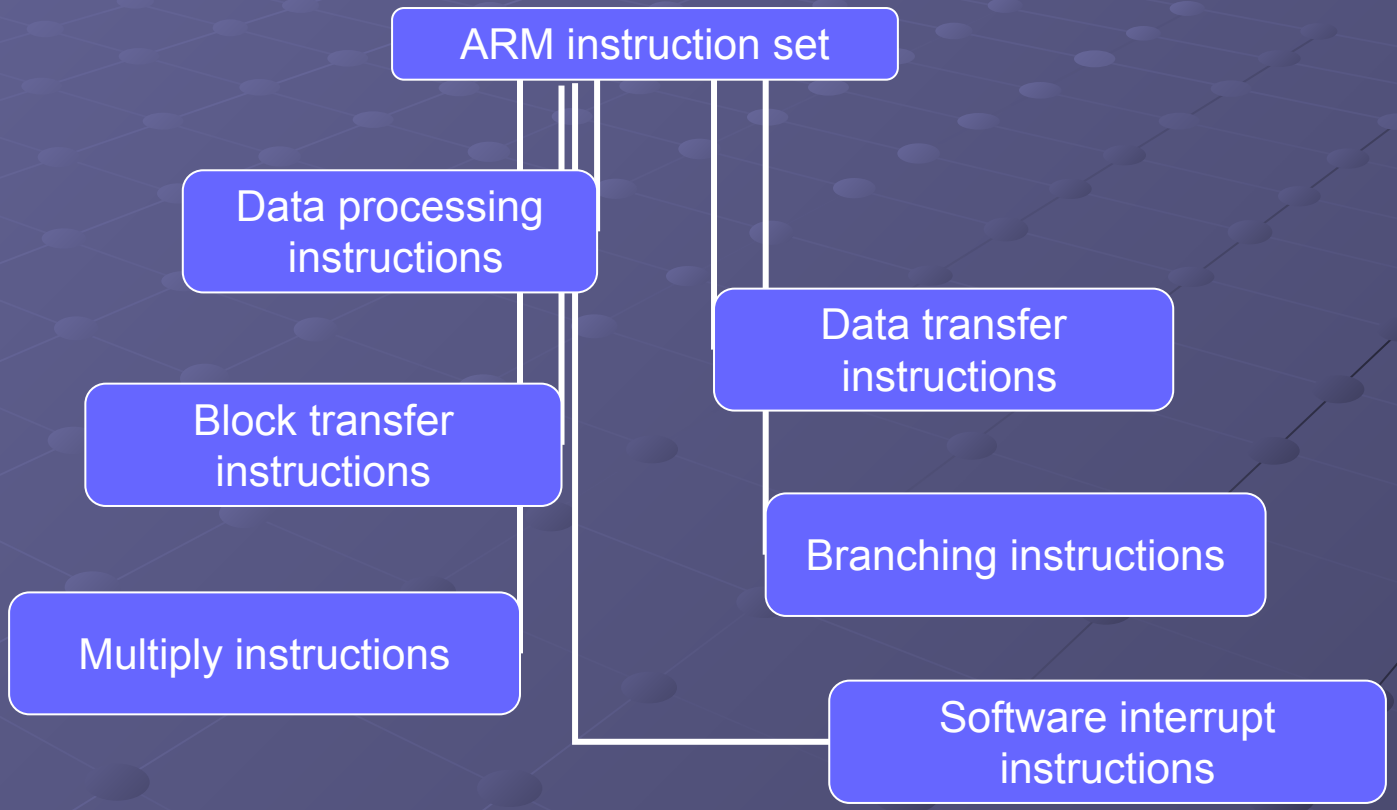
- Features:
  - Load/Store architecture
  - 3-address data processing instructions
  - Conditional execution
  - Load/Store multiple registers
  - Shift & ALU operation in single clock cycle

ARM Instruction Set (2)

- Conditional execution:
  - Each data processing instruction prefixed by condition code
  - Result – smooth flow of instructions through pipeline
  - 16 condition codes:

EQ	equal	MI	negative	HI	unsigned higher	GT	signed greater than
NE	not equal	PL	positive or zero	LS	unsigned lower or same	LE	signed less than or equal
CS	unsigned higher or same	VS	overflow	GE	signed greater than or equal	AL	always
CC	unsigned lower	VC	no overflow	LT	signed less than	NV	special purpose

ARM Instruction Set (3)



Data Processing Instructions

- Arithmetic and logical operations
- 3-address format:
  - Two 32-bit operands
    (op1 is register, op2 is register or immediate)
  - 32-bit result placed in a register
- Barrel shifter for op2 allows full 32-bit shift within instruction cycle

Data Processing Instructions (2)

- Arithmetic operations:
  - ADD, ADDC, SUB, SUBC, RSB, RSC
- Bit-wise logical operations:
  - AND, EOR, ORR, BIC
- Register movement operations:
  - MOV, MVN
- Comparison operations:
  - TST, TEQ, CMP, CMN

Data Processing Instructions (3)

Conditional codes

+

Data processing instructions

+

Barrel shifter

=

Powerful tools for efficient coded programs

Data Processing Instructions (4)

e.g.:

if (z==1) R1=R2+(R3*4)

compiles to

EQADDS R1,R2,R3, LSL #2

(SINGLE INSTRUCTION !)

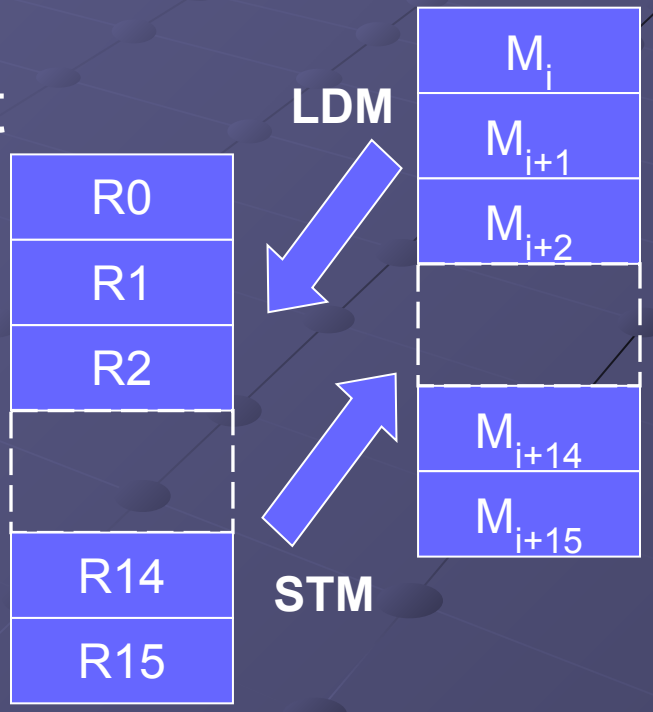
Data Transfer Instructions

- Load/store instructions
- Used to move signed and unsigned
  Word, Half Word and Byte to and from registers
- Can be used to load PC
  (if target address is beyond branch instruction range)

LDR	Load Word	STR	Store Word
LDRH	Load Half Word	STRH	Store Half Word
LDRSH	Load Signed Half Word	STRSH	Store Signed Half Word
LDRB	Load Byte	STRB	Store Byte
LDRSB	Load Signed Byte	STRSB	Store Signed Byte

Block Transfer Instructions

- Load/Store Multiple instructions (*LDM*/*STM*)
- Whole register bank or a subset copied to memory or restored with single instruction

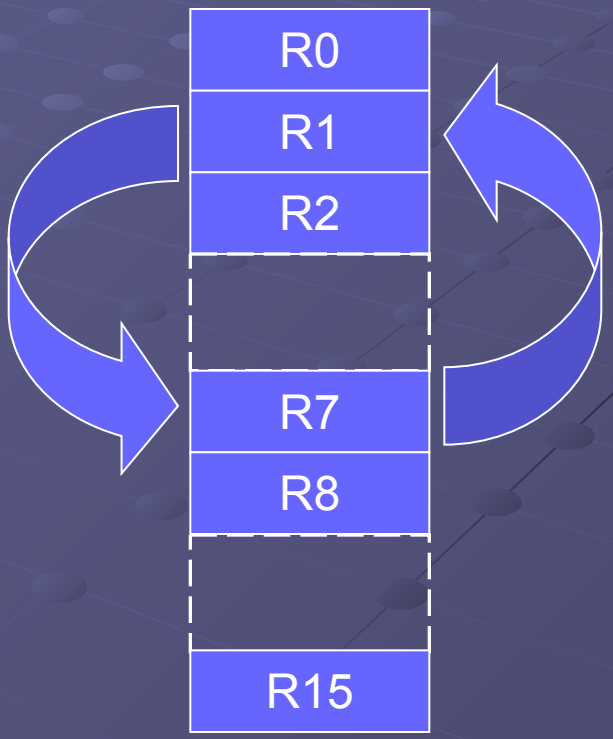


Swap Instruction

- Exchanges a word between registers
  - Two cycles

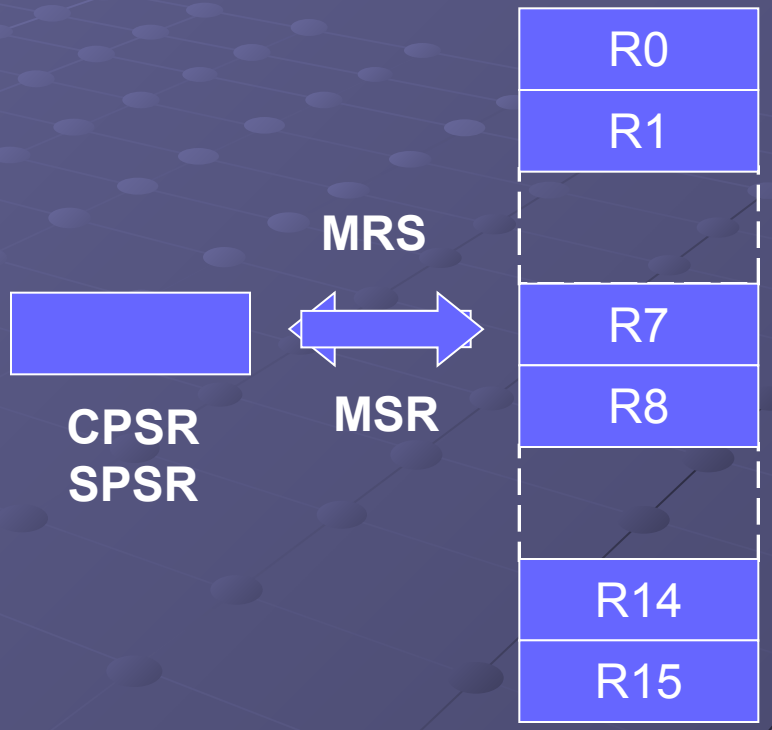
    but

    single atomic action
  - Support for RT semaphores



Modifying the Status Registers

- Only indirectly
- *MSR* moves contents from CPSR/SPSR to selected GPR
- *MRS* moves contents from selected GPR to CPSR/SPSR
- Only in privileged modes



Multiply Instructions

- Integer multiplication (32-bit result)
- Long integer multiplication (64-bit result)
- Built in Multiply Accumulate Unit (MAC)
- Multiply and accumulate instructions add product to running total

Multiply Instructions

- Instructions:

MUL	**Multiply**	**32-bit result**
MULA	**Multiply accumulate**	**32-bit result**
UMULL	**Unsigned multiply**	**64-bit result**
UMLAL	**Unsigned multiply accumulate**	**64-bit result**
SMULL	**Signed multiply**	**64-bit result**
SMLAL	**Signed multiply accumulate**	**64-bit result**

Software Interrupt

- *SWI* instruction
  - Forces CPU into supervisor mode
  - Usage: SWI #n

  - Maximum 2^{24} calls
  - Suitable for running privileged code and making OS calls

Branching Instructions

- *Branch* (B): jumps forwards/backwards up to 32 MB
- *Branch link* (BL): same + saves (PC+4) in LR
- Suitable for function call/return
- Condition codes for conditional branches

Branching Instructions (2)

- *Branch exchange (BX)* and *Branch link exchange (BLX)*: same as *B/BL* + exchange instruction set (ARM THUMB)
- Only way to swap sets

Thumb Instruction Set

- Compressed form of ARM
  - Instructions stored as 16-bit,
  - Decompressed into ARM instructions and
  - Executed
- Lower performance (ARM 40% faster)
- Higher density (THUMB saves 30% space)
- Optimal – "*interworking*" (combining two sets) – compiler supported

THUMB Instruction Set (2)

- More traditional:
  - No condition codes
  - Two-address data processing instructions
- Access to R0 – R8 restricted to
  - *MOV*, *ADD*, *CMP*
- PUSH/POP for stack manipulation
  - Descending stack (SP hardwired to R13)

THUMB Instruction Set (3)

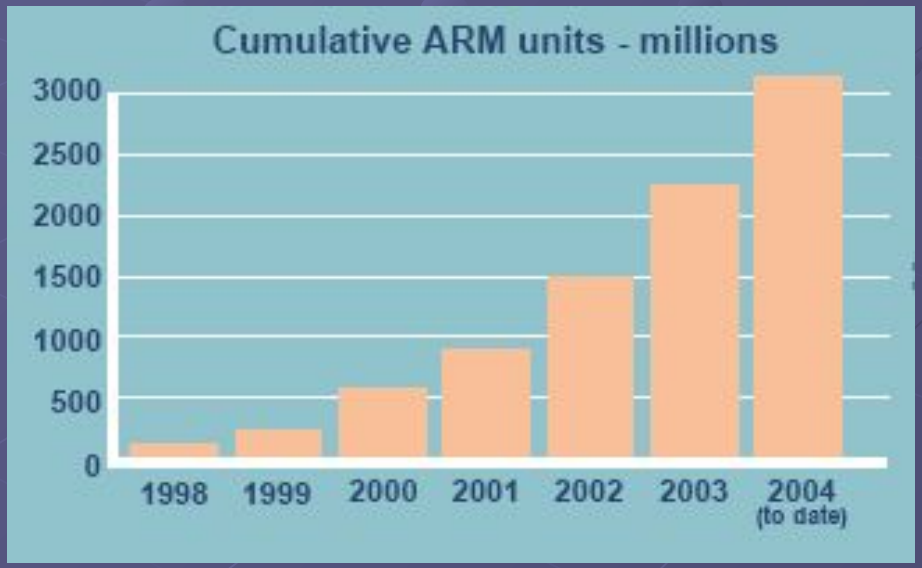
- No *MSR* and *MRS*, must change to ARM to modify CPSR (change using *BX* or *BLX*)
- ARM entered automatically after RESET or entering exception mode
- Maximum 255 SWI calls

The Next Step

- **New ARM Cortex family of processors**
  - New NEON™ media and signal processing extensions
  - Thumb®-2 blended 16/32-bit instruction set for performance and low power
  - Improved Interrupt handling

Summary

- Adoption of ARM technology has increased efficiency and lowered costs
- ARM is the world's leading architecture today
  - 3 billion ARM Powered chips and counting



References

- www.arm.com
- ARM Limited *ARM Architecture Reference Manual*, Addison Wesley, June 2000
- Trevor Martin *The Insiders Guide To The Philips ARM7-Based Microcontrollers,* Hitex (UK) Ltd., February 2005
- Steve Furber *ARM System-On-Chip Architecture (2nd edition),* Addison Wesley, March 2000

The End

Authors:
Nemanja Perovic, nemanjaizbg@yahoo.com
Prof. Dr. Veljko Milutinovic, vm@etf.bg.ac.yu