ARM Processor

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
http://csl.skku.edu
CPU Architecture
CPU & Memory

Memory

ADD r5, r1, r3

200

address

data

CPU

PC
von Neumann Architecture

- Separate CPU and memory
  - Memory holds data and instructions.
  - CPU fetches instructions from memory.
  - CPU manipulates data by performing arithmetic and logical operations

- CPU registers help out
  - Program Counter (PC)
  - Instruction Register (IR)
  - General-Purpose Registers (GPRs), etc.
Harvard Architecture

- **Data memory**
- **Program memory**

- **CPU**
  - PC

- **address**
- **data**
- **address**
- **instructions**
von Neumann vs. Harvard

- Harvard can’t use self-modifying code.
- Harvard allows two simultaneous memory fetches.
- Most DSPs use Harvard architecture for streaming data:
  - Greater memory bandwidth
  - More predictable bandwidth
RISC vs. CISC

- Complex Instruction Set Computer (CISC)
  - Many addressing modes
  - Many complex operations
  - Different instruction formats of varying lengths

- Reduced Instruction Set Computer (RISC)
  - Load/store
  - Pipelinable instructions
Instruction Set Architecture

- Supported operations
- Number of operands
- Types of operands
- Addressing modes
- Fixed vs. variable length
- Registers visible to the programmer
- ...
Multiple Implementations

- Successful architectures have several implementations:
  - Varying clock speeds
  - Different bus widths
  - Different cache sizes
  - ...
Assembly Language

- Textual description of instructions
- One instruction per line

- Pseudo-ops
  - Define current address, reserve storage, constants, ...

```
LDR r0, [r8]
ADD r4, r0, r1
CMP r4, r5 ; Compare r4 < r5
BGE label1

label1
```

Labels provide names for addresses
Comment
Introduction to
the ARM Architecture
Overview (1)

- Advanced RISC Machine
- The most widely used 32-bit ISA
  - 98% of the more than 1 billion mobile phones sold used at least one ARM processor (2005)
  - 4 billion shipped in 2008
  - 90% of all embedded 32-bit RISC processors (2009)
- ARM architecture has been extended over several versions.
Overview (2)

- ARM processors developed by licensees
  - DEC StrongARM
  - Marvell (formerly Intel) Xscale
  - Nintendo
  - TI OMAP
  - Qualcomm Snapdragon
  - Samsung Hummingbird/Exynos
  - Apple A4/A5/A5X/A6/A7
  - Nvidia Tegra
ARM Design Goals

- High performance
- Small code size
- Low power consumption
- Small silicon area
Application Processors

iPhone 3GS (600MHz)
iPhone 4 (clock speed not disclosed, 800MHz?)
iPad (1GHz)
Samsung Galaxy S (1GHz)
Samsung Galaxy Tab 7” (1GHz)

iPhone 4S (dual-core, 800MHz)
iPad 2 (dual-core, 1GHz)
New iPad (dual-core, 1GHz)
Galaxy S2 (dual-core, 1.2GHz)
Galaxy S3 (quad-core, 1.4GHz)

Galaxy S4 (quad-core 1.6GHz + quad-core 1.2GHz Cortex-A7)
Embedded Processors
Processors Lineup

Classic
ARM Processors

Application
Cortex Processors

Embedded
Cortex Processors

ARMv4T
ARMv5TJ
ARMv6
ARMv7A/R

ARM 32-Bit ISA

Thumb 16-Bit ISA

VFPv2
VFPv2
VFPv3

Jazelle
Jazelle
Jazelle

TrustZone
TrustZone

SIMD
SIMD

NEON

Virtualization

Cortex-A15
Cortex-A9
Cortex-A8
Cortex-A5
Cortex-A7
Cortex-A7
Cortex-M0

ARMv7M/ME
ARMv6M

SC300
SC000

Cortex-M3
Cortex-M1

Thumb
Thumb
ARM 7TDMI-S

- Implements the ARMv4T architecture
- Applications
  - iPod from Apple
  - Nintendo DS & Game Boy Boy Advance
  - Most of Nokia’s mobile phones
  - Lego Mindstorms NXT
  - iriver portable digital audio players
  - Roomba 500 series from iRobot
ARM Architecture (1)

- **RISC architecture**
  - A large uniform register file
  - A load/store architecture
  - Simple addressing modes
  - Uniform and fixed-length instruction fields
ARM Architecture (2)

- Other features
  - Arithmetic/logical operations combined with a shift
  - Conditional execution of almost all instructions
  - Auto-increment (-decrement) addressing modes
  - Load and Store Multiple instructions
ARM Registers

- 16 user-visible 32-bit registers: r0-r15
  - r13: Stack Pointer (SP)
  - r14: Link Register (LR)
  - r15: Program Counter (PC)

- Current Program Status Register (CPSR)
ARM Status Bits

- Every arithmetic, logical, or shifting operation sets CPSR bits:
  - N (Negative), Z (Zero), C (Carry), V (Overflow)
  - Condition codes updated when S bit = 1
- Example:
  - \( \text{0xffffffff} + \text{0x1} = \text{0x0} \); \( \text{NZCV} = 0110 \)
  - \( \text{0x7ffffff} + \text{0x1} = \text{0x80000000} \); \( \text{NZCV} = 1001 \)
  - \( \text{0x0} - \text{0x1} = \text{0xffffffff} \); \( \text{NZCV} = 1000 \)
Memory Model

- A linear collection of (up to $2^{32}$) bytes
- Supports both big-endian & little-endian
  - Controlled by the CFGBIGEND signal (ARM7)
  - CFGBIGEND == 0: Little-endian (default)
  - CFGBIGEND == 1: Big-endian
- Data types
  - word (32-bit): aligned to 4-byte boundary
  - halfword (16-bit): aligned to 2-byte boundary
  - byte (8-bit)
ARM Instructions

- Basic format
  - Two sources, one destination
  - All arithmetic operations have this form
    \[ \text{ADD } r0, r1, r2 \quad ; \quad r0 = r1 + r2 \]

- Operand types
  - Register (r0~r15):
    \[ \text{ADD } r0, r1, r2 \]
  - Immediate:
    \[ \text{ADD } r0, r1, \#4 \]
  - Memory:
    \[ \text{LDR } r5, [r3, \#32] \]
Thumb Instruction Sets (1)

- 16-bit instruction set
- A subset of the most commonly used 32-bit ARM instructions
- Significantly improved code density at a cost of some reduction in performance
  - Typically 65% of the ARM code size
  - 160% performance of the ARM code when running from a 16-bit memory system
Thumb Instruction Set (2)

- Transparently decompressed to full 32-bit ARM instructions in real time, without performance loss
- A processor executing Thumb instructions can change to ARM instructions for performance critical segments
- `#pragma thumb`
- `gcc -mthumb`
- `gcc -mthumb-interwork`
Thumb Registers

- r8 – r12 registers are not visible
- MOV, CMP, and ADD instructions can access high registers