Lecture 4 Introduction to the ARM Processor

How to improve on MU0?
- Larger address space than MU0 - 12-bit address gives 8k byte of memory. Therefore use 16-bit, 32-bit or wider address bus.
- Addition addressing modes - addressing modes are the different ways that the operand address may be specified. (See later)
- Subroutine call mechanism - this allows writing modular programs.
- Additional internal registers - this reduces the need for accessing external memory.
- Other system support mechanism such as interrupts, direct memory access, memory management etc.
- Allows coprocessors to be added (such as floating point processor).

Instruction formats
- 4-address instruction format (not used in ARM)
  \[ f \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \]
  \[ \text{function} \quad \text{op 1 addr} \quad \text{op 2 addr} \quad \text{dest. addr} \quad \text{next i addr} \]
- 3-address instruction format (used by ARM processor)
  \[ f \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \]
  \[ \text{function} \quad \text{op 1 addr} \quad \text{op 2 addr} \quad \text{dest. addr} \]
- 2-address instruction format (used by the Thumb instruction set of ARM)
  \[ f \text{ bits} \quad n \text{ bits} \quad n \text{ bits} \]
  \[ \text{function} \quad \text{op 1 addr} \quad \text{dest. addr} \]
- 1-address instruction format (used in MU0 and some 8-bit microcontrollers such as MC6811)
  \[ f \text{ bits} \quad n \text{ bits} \]
  \[ \text{function} \quad \text{op 1 addr} \]

How to make CPU faster?
- Wide instruction code and as few words (bytes) as possible
  - 8 bit / 16 bit / 32 bit / 64 bit processors
- Each instruction uses as few clock cycles as possible
- Keep as much data inside CPU as possible (many internal registers)
- Make each clock cycle as short as possible (high clock rate)
- Get each instruction to do as much as possible (?)
- What do you mean by “fast” anyway?

Design Approaches
- Complex Instruction Set Computers (CISC)
  - dense code, simple compiler
  - powerful instruction set, variable format, multi-word
  - multi-cycle execution, low clock rate
- Reduced Instruction Set Computers (RISC)
  - high clock rate, low development cost (?)
  - easy to move to new technology
  - simple instructions, fixed format, complex optimising compiler
- Fast local storage
  - Register file, on-chip memory, intelligent-RAM
- Concurrent execution of instructions
  - multiple function units - super scalar
  - “production line” arrangement - pipeline
- Direct hardware implementation
  - reconfigurable computing - no fetch/decode
A First Look at the ARM Processor

- **Main Features**
  - Load-Store architecture
  - Fixed-length (32-bit) instructions
  - 3-address instruction formats (2 source operand registers, 1 result operand register)
  - Conditional execution of ALL instructions
  - Multiple Load-Store register instructions
  - A single-cycle n-bit shift with ALU operation
  - Coprocessor instruction interfacing
  - Thumb architecture (dense 16-bit compressed instruction set)

The ARM Programmer's Model

- **R0 to R14 are general purpose registers (32-bits)**
  - Used by programmer for (almost) any purpose without restriction
- R15 is the Program Counter (PC)
- The remaining shaped ones are system mode registers - used during interrupts, exceptions or system programming (to be considered in later lectures)
- Current Program Status Register (CPSR) contains conditional flags and other status bits

### ARM CPSR format

<table>
<thead>
<tr>
<th>bit</th>
<th>31</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>Z</td>
<td>C</td>
<td>V</td>
<td>unused</td>
<td>I</td>
<td>F</td>
<td>T</td>
<td>mode</td>
<td></td>
</tr>
</tbody>
</table>

The ARM Programmer's Model (con't)

### ARM's memory organization

- **Maximum $2^{32}$ bytes of memory**
- A word = 32-bits, half-word = 16 bits
- Words aligned on 4-byte boundaries
- Half words aligned on even byte boundaries
Key features of the ARM instruction set

- Load-store architecture
- 3-address data processing instructions
- Conditional execution of EVERY instruction
- Inclusion of load and store multiple register instructions
- Perform a general shift operation and a general ALU operation in a single instruction executed in one cycle
- Can extend instruction set through the coprocessor instruction set, including adding new registers and data types
- Dense 16-bit compressed representation of instruction set in the Thumb architecture (to be consider much later)
- Combines the best of RISC and the best of CISC

ARM I/O System

- Handles all input/output peripherals (such as printer, disk and network) as memory-mapped devices
- Two types of interrupt support - normal interrupt and fast interrupt (considered in later lecture)
- Direct memory access (DMA) hardware support for high-bandwidth data transfer (also later)
- We will consider:
  - Parallel Peripheral Interface 82C55
  - Counter/Timer Interface 82C54
  - Universal Asynchronous Rx/Tx (UART) 16C450
  - Pulse Width Modulator
  - Interrupt Controller

ARM Development tools

- ARM C compiler - ANSI standard, fast, integrated
- ARM Assembler - translate assembly instructions to ARM instructions
- Linker
  - Takes one or more object files (from C compiler or ARM assembler) and combines them into one executable program
  - Resolve symbolic references (i.e. names of variables or routines are turned into actual memory addresses)
- ARM symbolic debugger - full control on execution and viewing of registers
- ARMulator - emulate the ARM processes with a system
  - Instruction-accurate modelling
  - Cycle-accurate modelling
  - Timing-accurate modelling
- Window User's interface