Overview

- Programmer’s Model of a Microprocessor
  - Address Space
  - Registers
  - Instruction Set
- Fetch – Decode – Execute Cycle
- Programmer’s Model of ARM 7TDMI
- Translation of C to ASM

Recall: Pre-Requisite

- Computers and Computing (e.g. COMP1011 & COMP1021)
  - C- Language Programming
  - The von Neumann model: memory/I/O/processing
  - The instruction set and execution cycle;
  - Registers and address spaces
  - An instruction set: operations and addressing modes
  - An expanded model of a computer: mass storage and I/O
  - The layered model of a computer: from gate-to-user-level

Review: What is Subject about?

- Coordination of many levels of abstraction
Review: Programming Levels of Representation

High Level Language Program (e.g., C)

Compiler

Assembly Language Program (e.g., ARM)

Assembler

Machine Language Program (ARM)

Machine Interpretation

temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;

ldr r0, [r2, #0]
ldr r1, [r2, #4]
str r1, [r2, #0]
str r0, [r2, #4]

1110 0101 1001 0010 0000 0000 0000 0000
1110 0101 1001 0010 0000 0000 0000 0100
1110 0101 1000 0010 0001 0000 0000 0000
1110 0101 1000 0010 0001 0000 0000 0100

ALUOP[0:3] <= InstReg[9:11] & MASK

Review: 5 Classic Components of a Computer

Control

Datapath

Memory

Processor

Input

Output

Network/Bus

ALU

Registers

An Expanded View of the Memory Systems

Processor

Control

Datapath

Register file

2nd Cache

Main Memory

Hard disk (Virtual Memory)

Speed: Fastest
Size: Smallest
Cost: Highest

• Cache is handled by hardware
• Virtual memory is handled by and Operating System
• Programmer sees only one memory and the registers

In ARM r15 (pc) is the program counter. It points to the instructions in memory

Registers

• Small and fast memory inside the processor
  - Load data from memory (Hold Data)
  - Store memory addresses (Hold Addresses)
  - Hold computation Operands and Results
  - Store back to memory

From memory

Selectors: 4 bits

Data: 32 bits

Register file: ARM has 16 Register: r0 - r15, (each 32 bits)

• There are other specialized registers as well which are not visible to the programmer
Fetch Decode Execute Cycle

Instruction Fetch
- Obtain instruction from program storage

Instruction Decode
- Determine required actions

Operand Fetch
- Locate and obtain operand data

Execute
- Compute result value or status

Result Store
- Deposit results in storage for later use

Next Instruction
- Determine successor instruction

The Programmer’s Model of a Microcomputer

Instruction Set:
- ldr r0, [r2, #0]
- add r2, r3, r4

Memory:
- 80000004 ldr r0, [r2, #0]
- 80000008 add r2, r3, r4
- 8000000B 23456
- 80000010 AEF0

Registers:
- r0 - r3, pc

Addressing Modes:
- ldr r12, [r1, #0]
- mov r1, r3

How to access data in registers and memory? i.e. how to determine and specify the data address in registers and memory

Memory Address Space (ARM 7TDMI)

$2^{30} =$ address space size in words
$4 \times 2^{30} =$ address space size in bytes = 4GBytes

Since 1980 almost every machine uses addresses to level of 8-bits (bytes)

16 Visible Registers (ARM 7TDMI)

They will be covered later

Memory mapped I/O
- 80000100 input
- 80000108 output

Programmer’s Model

Data 20hex = 3210 = 32

230 = address space size in words
4 x 230 = address space size in bytes = 4GBytes

A word (4 bytes in memory)

Since 1980 almost every machine uses addresses to level of 8-bits (bytes)

Memory:
- 80000004 ldr r0, [r2, #0]
- 80000008 add r2, r3, r4
- 8000000B 23456
- 80000010 AEF0

Registers:
- r0 - r3, pc

Addressing Modes:
- ldr r12, [r1, #0]
- mov r1, r3

How to access data in registers and memory? i.e. how to determine and specify the data address in registers and memory

Memory Address Space (ARM 7TDMI)

Addresses
- 0x80000000
- 0x80000004
- 0x80000008
- 0x8000000C

Binary Contents
- E0832004
- E0452006
- E8920000
- 0x00000020

Data 20hex = 32

230 = address space size in words
4 x 230 = address space size in bytes = 4GBytes

Since 1980 almost every machine uses addresses to level of 8-bits (bytes)

addresses
- 0x80000000
- 0x80000004
- 0x80000008
- 0x8000000C

interpretations

addresses
- 0x80000000
- 0x80000004
- 0x80000008
- 0x8000000C

Binary Contents
- E0832004
- E0452006
- E8920000
- 0x00000020

Data 20hex = 32

230 = address space size in words
4 x 230 = address space size in bytes = 4GBytes

Since 1980 almost every machine uses addresses to level of 8-bits (bytes)

addresses
- 0x80000000
- 0x80000004
- 0x80000008
- 0x8000000C

Binary Contents
- E0832004
- E0452006
- E8920000
- 0x00000020

Data 20hex = 32

230 = address space size in words
4 x 230 = address space size in bytes = 4GBytes

Since 1980 almost every machine uses addresses to level of 8-bits (bytes)
**Instruction Set (ARM 7TDMI)**

- Set of instruction that a processor can execute
- Instruction Categories
  - Data Processing or Computational (Logical and Arithmetic)
  - Load/Store (Memory Access: or transferring data between memory and registers)
  - Control Flow (Jump and Branch)
  - Floating Point
  - coprocessor
  - Memory Management
  - Special

<table>
<thead>
<tr>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
</tr>
<tr>
<td>r1</td>
</tr>
<tr>
<td>r2</td>
</tr>
<tr>
<td>r3</td>
</tr>
<tr>
<td>r4</td>
</tr>
<tr>
<td>r5</td>
</tr>
<tr>
<td>r6</td>
</tr>
<tr>
<td>r7</td>
</tr>
<tr>
<td>r8</td>
</tr>
<tr>
<td>r9</td>
</tr>
<tr>
<td>r10</td>
</tr>
<tr>
<td>r11</td>
</tr>
<tr>
<td>r12</td>
</tr>
<tr>
<td>r13</td>
</tr>
<tr>
<td>r14</td>
</tr>
<tr>
<td>r15(PC)</td>
</tr>
</tbody>
</table>

**Data Processing Instructions**

- Data Processing Instructions:
  - operate ONLY on registers
  - store result ONLY on registers
  - Category: Arithmetic, Logical, Data movement
- Examples:
  - `mov r1, r2 ; r1 ← r2`
  - `add r1, r2, r3 ; r1 ← r2 + r3`
  - `and r3, r3, r4 ; r3 ← r3 AND r4`

<table>
<thead>
<tr>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
</tr>
<tr>
<td>r1</td>
</tr>
<tr>
<td>r2</td>
</tr>
<tr>
<td>r3</td>
</tr>
<tr>
<td>r4</td>
</tr>
<tr>
<td>r5</td>
</tr>
<tr>
<td>r6</td>
</tr>
<tr>
<td>r7</td>
</tr>
<tr>
<td>r8</td>
</tr>
<tr>
<td>r9</td>
</tr>
<tr>
<td>r10</td>
</tr>
<tr>
<td>r11</td>
</tr>
<tr>
<td>r12</td>
</tr>
<tr>
<td>r13</td>
</tr>
<tr>
<td>r14</td>
</tr>
<tr>
<td>r15(PC)</td>
</tr>
</tbody>
</table>

**Memory Access Instructions**

- Memory Access Instructions:
  - Transfer data from a memory address to a register (load instructions)
  - Transfer data from a register to a memory address (store instructions)
- Examples:
  - `ldr r1, [r2] ; r1 ← mem[r2]
  - `str r1, [r3] ; r1 → mem[r3]

**Control Flow Instructions**

- Control Flow Instruction:
  - Generally next Instructions are fetched from Sequential addresses in Mem
  - Some Instructions cause fetch of next instruction from non sequential addresses in Mem (Control flow or branch instructions)
- Examples:
  - `br there`

<table>
<thead>
<tr>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80000000</td>
</tr>
<tr>
<td>0x80000004</td>
</tr>
<tr>
<td>0x80000008</td>
</tr>
<tr>
<td>0x8000000C</td>
</tr>
<tr>
<td>0x80000018</td>
</tr>
</tbody>
</table>

All will be covered in detail later.
What's this stuff good for? GameBoy!

- **Nintedo Micro GameBoy**
  - The Micro, which began shipping September 2005, is easily the smallest member of Nintendo's hit GameBoy family.
  - The device is true to its name, measuring just 4 inches long by 2 inches deep and 0.68 inches thick.
  - The GameBoy Micro is smaller than a deck of cards yet packs gaming power.
  - Nintendo says it has about the same power of previous GameBoy models despite its diminutive size.
  - It includes a 2-inch color LCD and buttons for both game control and menu selection. Nintendo also markets a variety of face plates to change to look of the device.
  - Powered by ARM Processor
  - USD100


Computers In the News!

- **ARM7 and Nucleus RTOS on Tour with Paul McCartney**

  The Clair iO mastering processor (designed with an ARM7 core from Lake Technology Limited, Sydney, Australia) has been used in tours featuring Paul McCartney and other top artists.

  The Clair iO is a 2-input, 6-output loudspeaker controller that employs 40-bit floating point DSP processing for a wholly innovative approach to live sound.

  The iO's innovative design is unique in its wireless network capability. The Nucleus Real-Time Operating System (RTOS) was used to develop a wireless DSP loudspeaker controller used by audio engineers to control live sound quality and management in concert arenas.

  Within the Clair iO processor, Nucleus acts as the communications link between the various host controllers on the Ethernet control side and the DSP processors, which manipulate the audio, on the other side. The iO processor is designed so that the ARM processor running the RTOS is separated from the DSP function.


ELEC2041 Reading Materials (#2/2)

- **Textbooks:**
  - Main references for lecture material:
    - Steve Furber: ARM System on-chip 2nd Ed, Addison-Wesley, 2000, ISBN: 0-201-67519-6. We use chapters 2, 3, 5 and 6, 8, 9, 10, & 11
  - Additional references for lectures and labs:
    - David Patterson and John Hennessy: Computer Organisation & Design: The HW/SW Interface," 2nd Ed 1996. Relevant chapters are, 3, 4 & 8
  - C-Programming

ELEC2041 Laboratory Schedule

- **Laboratory:**
  - Monday: 09:00 – 11:00 EE233
  - Monday: 12:00 – 14:00 EE233
  - Tuesday: 15:00 – 17:00 EE233
  - Thursday: 09:00 – 11:00 EE233
  - Thursday: 12:00 – 14:00 EE233
  - Thursday: 15:00 – 17:00 EE233
  - Friday: 12:00 – 14:00 EE233
  - Friday: 15:00 – 17:00 EE233

  You will be only allowed into the lab class that you are enrolled in. No exception allowed.

  - All Lab Classes Start from Week #3
  - There is a Possibility of Starting Special Open Access labs
  - Wednesday: 17:00 – 19:00 EE233
  - Thursday: 17:00 – 19:00 EE233
  - Not assessed
  - It is for those who need a bit of extra time
Laboratory Groups

- Linux Lab Group Account
  - Day       Time       Group User Name
  - Monday: 09:00 – 11:00  ea01 – ea15
  - Monday: 12:00 – 14:00  eb01 – eb15
  - Tuesday: 15:00 – 17:00 ec01 – ec15
  - Thursday: 09:00 – 11:00 ed01 – ed15
  - Thursday: 12:00 – 14:00 ee01 – ee15
  - Thursday: 15:00 – 17:00 ef01 – ef15
  - Friday: 12:00 – 14:00  eg01 – eg15
  - Friday: 15:00 – 17:00  eh01 – eh15

**PASSWORD: group_xxxx**
With xxxx being the group number, eg group_ea01, group_ee01
You must change your password the first time you log in.

Laboratory Format

- In group of two partners
- You choose your partner in Sign Up Class (Week #2 for Friday classes, Week #3 all other classes). It CANNOT be changed later
- You will get a group Linux Account
- No formal report to hand in
- You are assessed based on a system of checkpoints

<table>
<thead>
<tr>
<th>Checkpoint 3:</th>
<th>Signature:</th>
</tr>
</thead>
</table>

Assemble, link and run your program using the GNU Tools. Show your working program to the Laboratory Assessor.

- Assessors mark you check points
- Lab Demonstrators help you with the labs
- Extra Credit Checkpoints: For those who want to do more for bounce marks (max marks) (accepted if you have already finished the normal checkpoints)

ELEC2041 Software

- Edit Utility Tools
  - Enable creation of C or assembly source programs for ARM Processor on a Linux Platform

- GNU ARM Cross Compiler and Assembler Tools:
  - Enable Translation by Compilation, Assembly, and Linking of source programs into ARM object programs; Executable and Linking Format (ELF)

- GNU ARM Source Level Debugger
  - Enables simulation of ARM ELF programs while referencing back to the source code.

- Komodo Integrated Debugger
  - Enables downloading of ARM ELF code into the target ARM Processor on DSLMU Development Board
  - Enables Execution and debugging of the downloaded program on the target processor on DSLMU Development Board

All Tools included in the Companion CD-ROM

Laboratory Documentation

- Written Extensively
- They Server as:
  - Lecture Notes
  - Tutorials
  - AND Practical exercise

- Careful Reading Enables you to:
  - Understand the Subject material
  - Do tutorial practice
  - AND get practical experience

DO TAKE THEM VERY SERIOUS!
Sample Assembly Program

C statement: \( k = k - 2 \)

Binary Contents

```
E3A02094
    mov r2, #0x94
E3A05002
    mov r5, #2
E5920000
    ldr r0, [r2]
E5820000
    str r0, [r2]
```

Location for variable \( k \)

Compilation

- How to turn notation programmers prefer into notation computer understands?
- Program to translate C statements into Assembly Language instructions; called a compiler
- Example: compile by hand this C code:
  
  \[
  a = b + c;
  d = a - e;
  \]
- Easy: add r1, r2, r3
  sub r4, r5, r6
- Big Idea: compiler translates notation from 1 level of abstraction to lower level

Conclusion

- ARM has 16 32-bit registers
- Instructions are all 32 bits
- Instruction Categories
  - Data Processing or Computational (Logical and Arithmetic)
  - Load/Store (Memory Access: or transferring data between memory and registers)
  - Control Flow (Jump and Branch)
- Access to memory is only through \texttt{ldr} and \texttt{str} instructions