ELEC2041
Microprocessors and Interfacing

Lecture 5: Programmer's Model of Microprocessors

http://webct.edtec.unsw.edu.au/

March 2005

Saeid Nooshabadi

saeid@unsw.edu.au

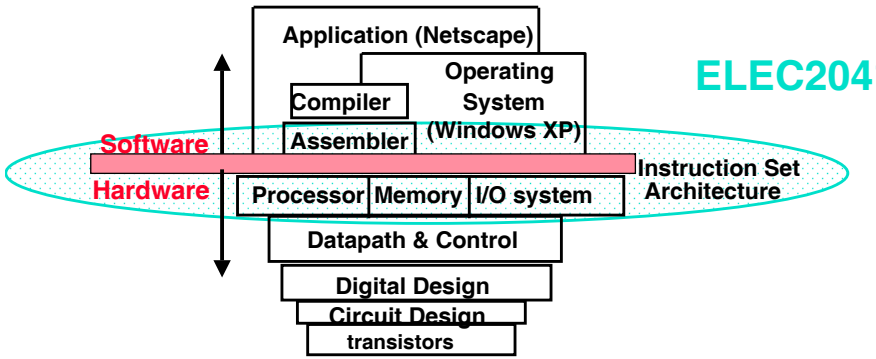
Overview

- 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

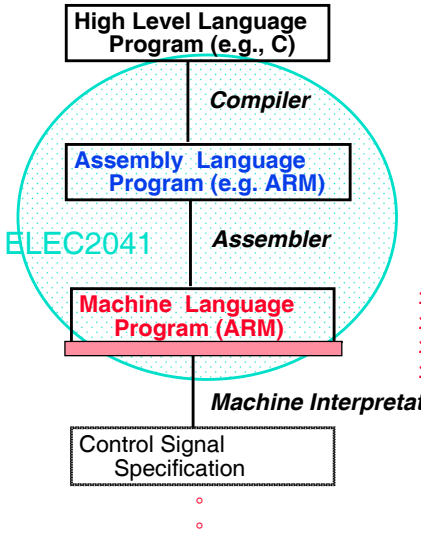
- Computers and Computing (*e.g.* COMP1011 & COMP1021)
  - C- Language Programming
  - The von Neumann model: memory/I-O/processing
  - The instruction set and execution cycle;
  - Registers and address spaces
  - An instruction set: operations and addressing modes
  - An expanded model of a computer: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



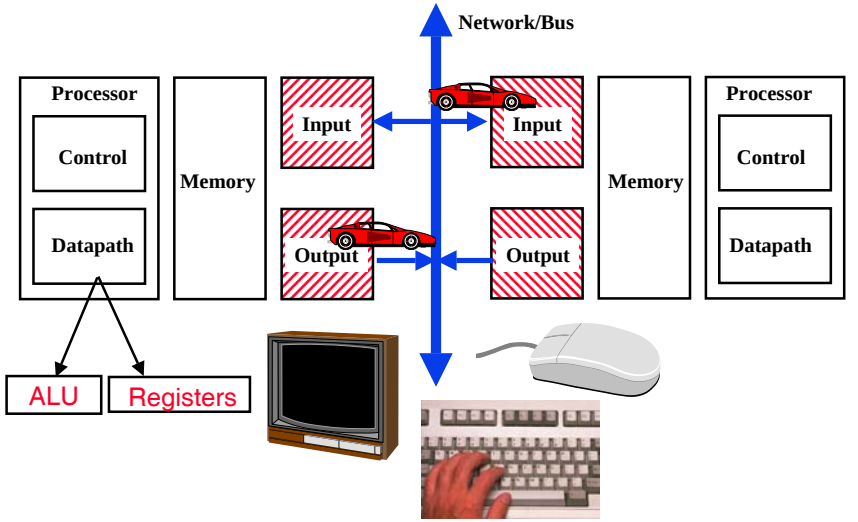

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

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

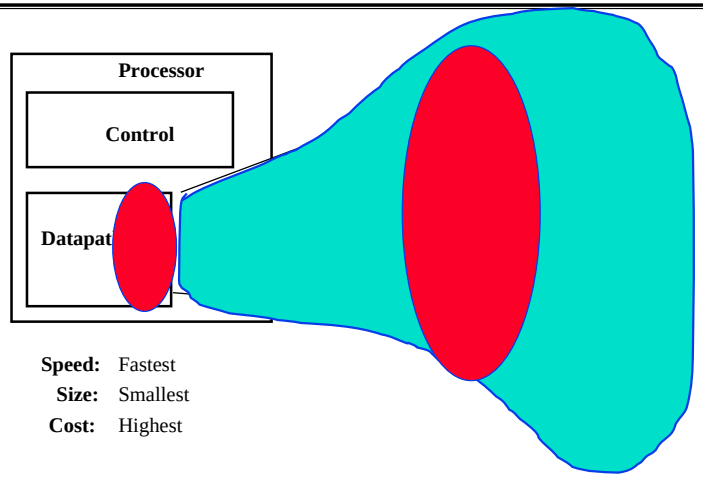
```
1110 0101 1001 0010 0000 0000 0000 0000
1110 0101 1001 0010 0000 0000 0000 0100
1110 0101 1000 0010 0001 0000 0000 0000
1110 0101 1000 0010 0001 0000 0000 0100
```

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

Review: 5 Classic Components of a Computer



An Expanded View of the Memory Systems

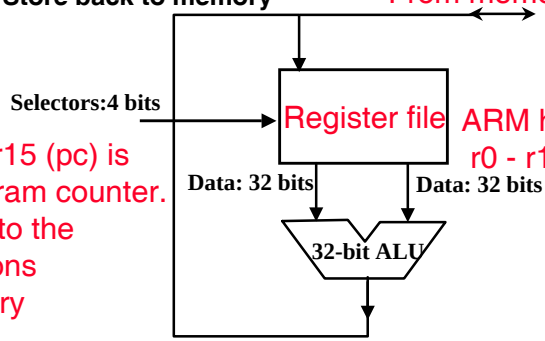


Speed: Fastest
Size: Smallest
Cost: Highest

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

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**

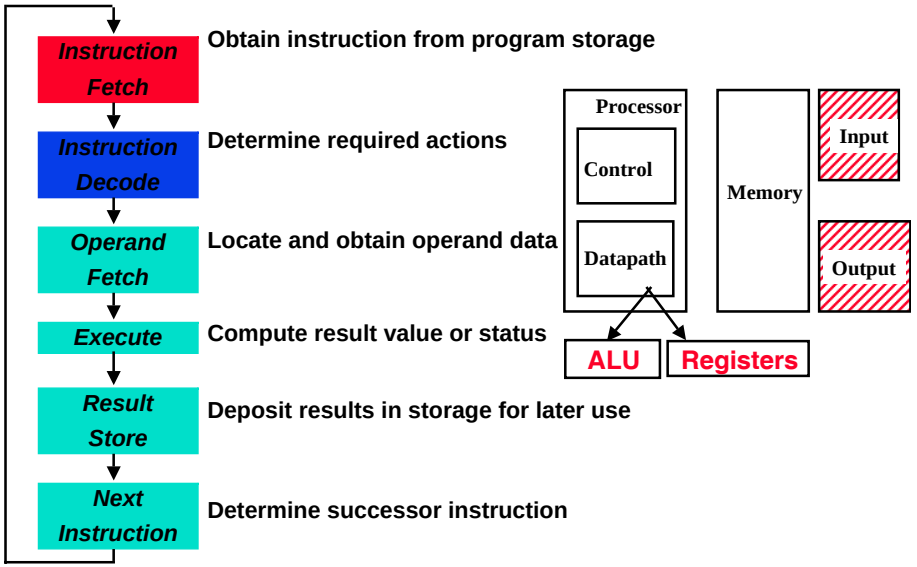


ARM has 16 Register: r0 - r15, (each 32 bits)

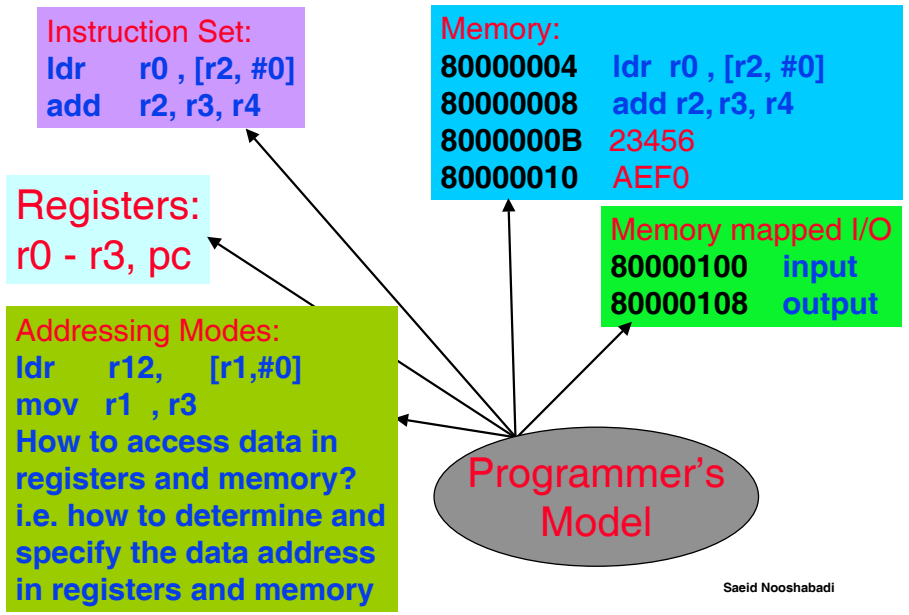
In ARM r15 (pc) is the program counter. It points to the instructions in memory

- **There are other specialized registers as well which are not visible to the programmer**

Fetch Decode Execute Cycle



The Programmer's Model of a Microcomputer



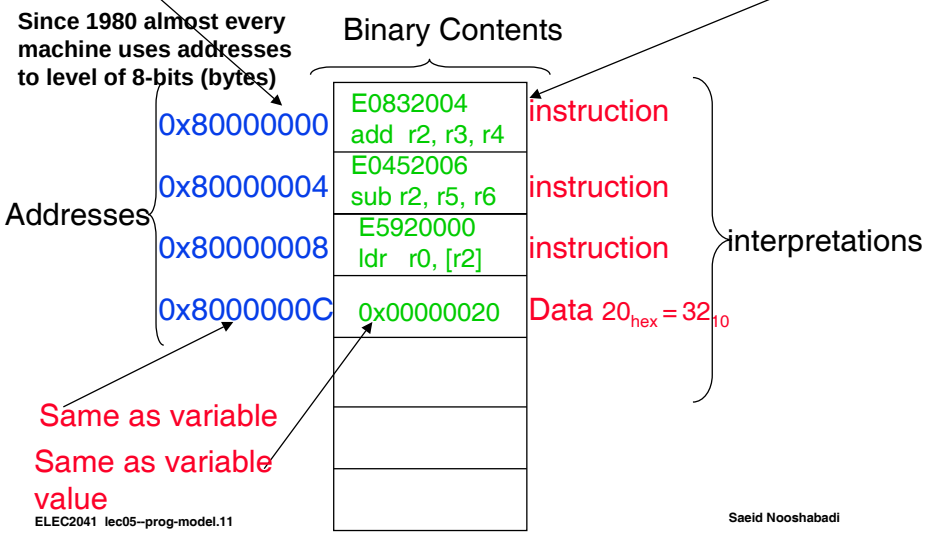
Memory Address Space (ARM 7TDMI)

2^{30} = address space size in words

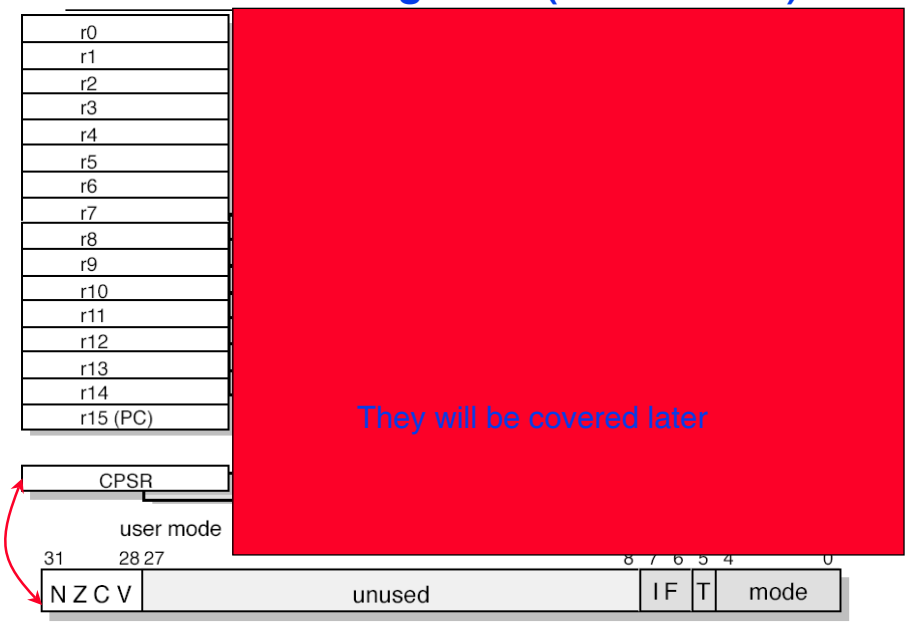
4 x 2^{30} = address space size in bytes = 4GBytes

A word (4 bytes in memory)

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

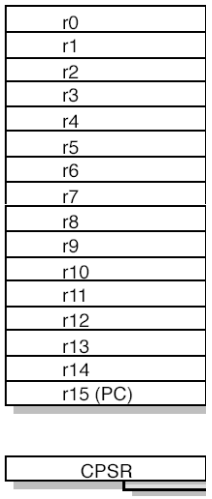


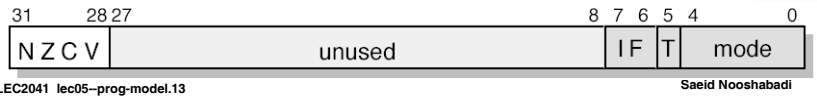
16 Visible Registers (ARM 7TDMI)



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



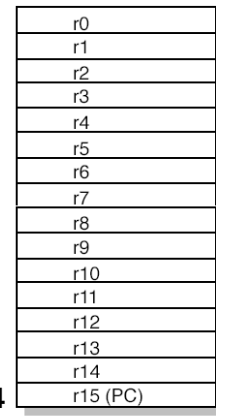


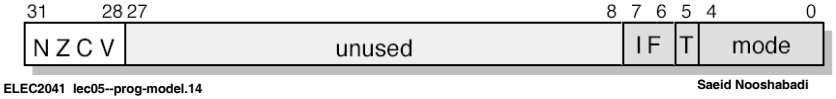
Data Processing Instructions

- 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
```

All will be covered in detail later





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]
```

Address of memory location is in register r2

```
str r1, [r3]   ; r1 → mem[r3]
```

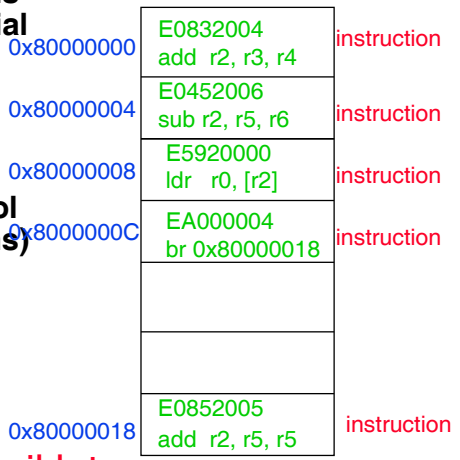
Address of memory location is in register r3

All will be covered in detail later

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
```



All will be covered in detail later

What's this stuff good for? GameBoy!

° Nidendo Executive GameBoy
- Power by ARM Processor
- Color LCD: 240 x 160 pixel (32 000 colors)
- USD100

Is This the Executive's GameBoy?

Nintendo's newest handheld GameBoy ditches the just-a-toy look in favor of a clamshell design that an adult player might use in public without embarrassment. The GameBoy Advance SP has the appearance of a smallish DVD player, and, except for its shape and a couple of extras, has the same features as the GameBoy Advance, introduced in June 2001.

Remedying a complaint about its older model, Nintendo has added a front light to the color screen to make playing easier in poor lighting. Also, the older unit's pair of AA alkaline batteries didn't last long. So this time around, Nintendo powers the unit with a rechargeable lithium-ion battery good for 10 hours of play, with a three-hour recharge period; switch off the front light, and it's good for 18 hours.

Nintendo's GameBoy Advance SP was unveiled last month in Tokyo. The silver version promises to please the jet-setting executive who has everything.

Like its predecessor, the SP has a 2.9-inch reflective thin-film transistor LCD screen and is powered by a zippy 32-bit ARM microprocessor. It has the same resolution—240 by 160 pixels—and displays some 32 000 colors. But it can be folded like a laptop to take up less space on your palm than a 3.5-inch floppy disk: around 8.5 by 8.2 cm by less than 2.54 cm in height. It weighs about 140 grams. Nintendo styles its colors as metallic cobalt, a blue, or, for that executive look, a high-end–looking platinum, more usually called silver.

Nintendo promises that all current and future games for GameBoy Advance—and the list is already huge—will be compatible with the SP. The unit will list for ¥12 500 in Japan when it becomes available on 14 February, and for US $99.95 in the United States when it bows there on 23 March.

For more information, see Nintendo's Web site at http://www.nintendo.com, or order from GameBoy vendor EBgames.com at http://www.ebgames.com/. ●

IEEE Spectrum Feb 2003

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.

http://www.convergencepromotions.com/IQ/issue5/iss5(Pg58-59).pdf

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
  - Waldron, John: Introduction to RISC Assembly Language, Addison-Wesley Publishing, 1999, ISBN: 0201398281.
- C-Programming
  - Brian Kernighan & Dennis Ritchie: The C Programming Language, 2nd Ed., Prentice Hall, 1988, ISBN:0-13-110362-8

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.
- Starts from week #2 for Friday Classes and Week #3 for all other classes
- 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

> Assemble, link and run your program using the GNU Tools. Show your working program to the Laboratory Assessor.
>
> Checkpoint 3: Signature:

- 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

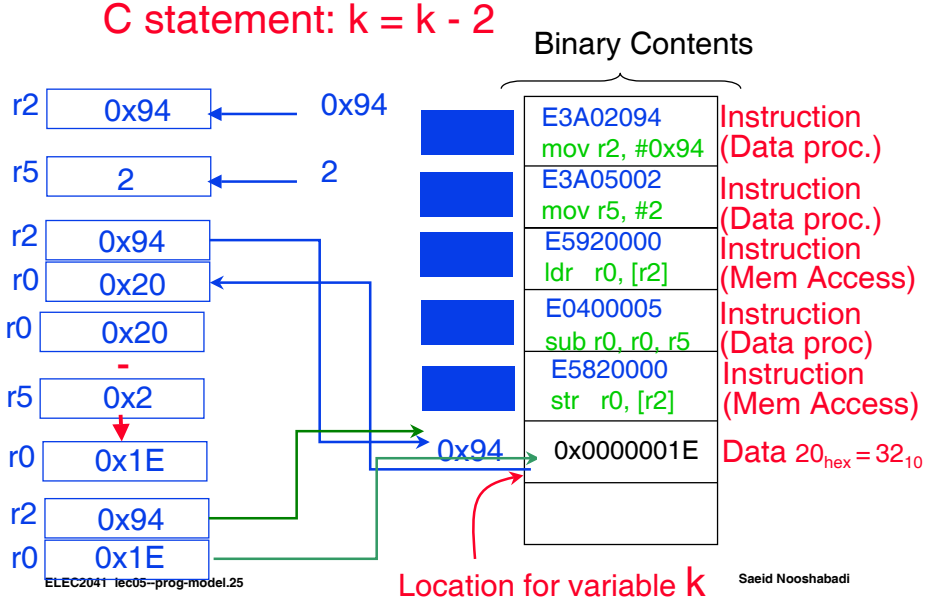
All Tools included in the Companion CD-ROM

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



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 `ldr` and `str` instructions