“The performance of future software systems will be dramatically affected ... by how well software designers understand the basic hardware techniques at work in a system”

David A. Patterson and John L. Hennessy

“A person who is more than casually interested in computers should be well schooled in machine language, since it is a fundamental part of a computer.”

Donald E. Knuth
On successful completion of 3D1 you will be able to:

- describe the basic characteristics, structure and operation of a microprocessor system;
- translate between simple high-level programming language constructs and their assembly language equivalents;
- design, construct, document and test small-scale assembly language programs to solve simple problems;
- reason about the cost of executing instructions and the efficiency of simple programs;
- make use of appropriate documentation and reference material.
ARM7TDMI

- iPod, Nintendo DS, Nokia mobiles, Lego Mindstorms, ...
- NXP LPC2468 32-bit **microcontroller**
  - ARM7TDMI-S CPU
  - Flash memory (512KiB), RAM (96KiB)
  - 10/100 Ethernet, USB 2.0, A/D & D/A converters, ...
Development Environment

```
GLOBAL Reset_Handler
AREA Reset, CODE, READONLY
ENTRY
Reset_Handler
main
ADR r4, teststr
MOV r5, r4
loop1
LDRB r7, [r5]
CMP r7, #0
BEQ eloop1
ADD r5, r5, #1
B loop1

*** Currently used: 112 Bytes (0%)
SetupForStart(); // Setup for Run
```

*** error 34, line 104: undefined identifier
Demonstration

- Keil µVision Development Environment
- Writing a simple program
- “Building” the program
- Loading the program into memory and debugging it
- Observing the results
A simple program that adds four numbers

- Make the first number our subtotal
- Add the second number to the subtotal
- Add the third number to the subtotal
- Add the fourth number to the subtotal
Demonstration

```
AREA Demo, CODE, READONLY
IMPORT main
EXPORT start

.start
   MOV r0, r1 ; Make the first number the subtotal
   ADD r0, r0, r2 ; Add the second number to the subtotal
   ADD r0, r0, r3 ; Add the third number to the subtotal
   ADD r0, r0, r4 ; Add the fourth number to the subtotal
stop B stop
.END
```

*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 416 Bytes (1%)

SetupForStart(); // Setup for Running
start // Goto Main

ASSIGN BreakDisable BreakEnable BreakKill BreakList

Real-Time Agent: Target Stopped  ULINK ARM Debugger  ti: 0.00000000 sec  Li6
Program 1.1 – Demonstration

AREA Demo, CODE, READONLY
IMPORT main
EXPORT start

start
  MOV r0, r1 ; Make the first number the subtotal
  ADD r0, r0, r2 ; Add the second number to the subtotal
  ADD r0, r0, r3 ; Add the third number to the subtotal
  ADD r0, r0, r4 ; Add the fourth number to the subtotal

stop B stop

END
Reading – ARM Assembly Language

- **Recommended reading**

- **Other reading**
  - Peter Knaggs, Stephen Welsh, *ARM: Assembly Language Programming*, Bournemouth University, 2004
Reading – Computer Architecture

- Other reading
Simple Model of a Microprocessor System

- **A Processing Unit** which performs operations on data
- **Memory**, which stores:
  - **Data**: representing text, images, videos, sensor readings, \( \pi \), audio, etc. ...
  - **Instructions**: **Programs** are composed of sequences of instructions that control the actions of the processing unit
- Instructions typically describe very simple operations, e.g.
  - **Add** two values together
  - **Move** a value from one place to another
  - **Compare** two values
Memory is arranged as a series of "locations"

Each location has a unique "address"
- e.g. the memory location at address 21000 contains the value 49

The number of locations in memory is limited
- e.g. 2GB of RAM ⇒ 2,147,483,648 locations!

Each location can contain either data or an instruction

Instructions are encoded as values
- e.g. the value 49 might be the code used to tell the processor to add two values together
Program execution

- When the computer is turned on, the processing unit begins executing the instruction in memory at the address stored in the Program Counter or PC.

- After executing an instruction, the value of the Program Counter is changed to the address of the next instruction in the program.

- The processing unit keeps doing this until the computer is turned off.
This simple model of a programmable computer is the model used by computers familiar to us (PCs, games consoles, mobile phones, engine management units, ...)

Behaviour is entirely predictable (**deterministic**)

*If that’s the case, how can computers generate random numbers?*

The “power” of computers arises because they perform a lot of simple operations very quickly

The complexity of computers arises because useful programs are composed of many thousands or millions of simple instructions

*Possibly executing in parallel on more than one computer!*