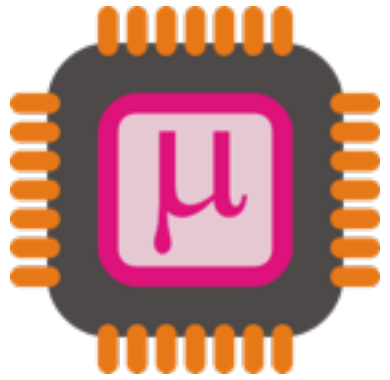
Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

01 – Introduction

CS1021 – Introduction to Computing I

Dr Jonathan Dukes | jdukes@tcd.ie
School of Computer Science and Statistics

You will be able to …



describe the basic characteristics, structure and operation of a simple **microprocessor** (or microprocessor system)

represent / interpret basic information (numbers, text) in **binary** form

01101100
01101111
01110110
01100101

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

jdukes@tcd.ie

Motivation

"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

jdukes@tcd.ie

Some Important Housekeeping

Lectures

Monday 2PM in Hamilton MacNeill (H3)

Tuesday 11AM in LB01

Tutorials

Thursday 9AM in Goldsmith Hall <u>**or**</u>

1PM in LB04

Check which tutorial you should attend on Blackboard

No tutorial in Week 1

Labs

Friday 10AM, 11AM, 12 noon or 1PM in LG35/LG36 (O'Reilly Institute)

There will be labs in Week 1

Check which lab you should attend on Blackboard (available Thursday)

Again!

No Tutorial in Week 1

There will be Labs in Week 1

Check Blackboard for Tutorial and Lab Time
http://mymodule.tcd.ie – CS1021 Introduction to Computing

jdukes@tcd.ie

Assessment

Labs	four, working in pairs, with a new exercise every fortnight	**30%** continuous assessment
Assignments	two, working individually	
Examination	2 hours	**70%** exam

5 credits (out of 60 for the full year of your degree course)

*(and another 5 for **CS1022 Introduction to Computing II** for most of you!)*

Jonathan Dukes | jdukes@tcd.ie

Attendance and Deadline Regulations

Attendance at all lectures, labs and tutorials is compulsory

As a practicality … catch up as quickly as you can (e.g. by working through lecture notes, tutorials and lab exercises in your own time)

- obtain material not available on-line from other students
- inform your tutor if you (will) miss a major deadline or are absent for more than one consecutive lecture/lab/tutorial

Zero marks for late coursework without explanation

You may be returned as <u>Non-Satisfactory</u> (see College Calendar) if you miss more than one-third of your Lectures, Labs or Tutorials, if you fail to submit more than one third of your Lab Exercises or if you fail to submit either Assignment.

jdukes@tcd.ie

Your Own Work

When you submit work as part of an exercise or assignment, you are implying that it is <u>your own work</u>

DO indicate where you received help from someone other than a lecturer, teaching assistant or demonstrator

DO indicate where you have used other sources of information (e.g. websites or text books)

DON'T share your work with other students – in your year or any other year – you will make it harder for them to succeed in College

DO discuss your work with each other, ask another student for hints to solve problems, ask for assistance fixing bugs, etc.

DO be prepared to explain any work that you submit and expect us to use plagiarism detection tools such as TurnItIn

Taking credit for someone else's work without giving them due recognition is a serious academic offence ("plagiarism", see your *Course Handbook* and the *College Calendar*)

jdukes@tcd.ie

Recipe for Success

Do …

go back over each lecture, particularly any examples, **adding your own notes**

all of the coursework (tutorials, labs, assignments) **yourself**

Don’t wait for someone to …

come looking for **coursework that you haven’t submitted**

tell you that you **don’t understand** something

ask you why you **haven’t been attending**

If you think you are falling behind …

spend some time **studying** the problem

revise the lecture slides, examples, tutorials, labs

get **help** from **classmates**, **demonstrators** or **lecturer** if necessary

Don’t think that if you don’t understanding something now you can fix it before the exam … **this module doesn’t work that way** … I have ample evidence of this!

Laptops etc.

Laptops, tablets, phones etc. **may not** be used during lectures

There are **exceptions** (e.g. if you are registered with the College Disability Service and require the use of a laptop, just let me know)

You may use a laptop / tablet during tutorials **for referring to course material**, documentation, etc. only

You may use your laptop / tablet during labs

Mueller, Pam A.; Oppenheimer, Daniel M., "The Pen Is Mightier Than the Keyboard: Advantages of Longhand Over Laptop Note Taking," Psychological Science, June 2014, Vol. 25, No. 6. doi: 10.1177/0956797614524581.

jdukes@tcd.ie

ARM Architecture

The microprocessor system we will study is based on the ***ARM architecture*** (contrast with *Intel 64*, *AMD64*, *IA-64*, etc.)

jdukes@tcd.ie

Rivals Intel and Arm partner for new chip manufacturing venture

British chip designer Arm has announced it will license its technology for a new manufacturing venture with longtime rival Intel, as the world's biggest chipmaker continues to adjust to a post-PC world.

The deal with Arm, which will be acquired by Japanese company SoftBank with effect from September 5, gives Intel another way into the booming smartphone and internet of things markets, where it has struggled to grab semiconductor share from Arm-based chipmakers.

Financial Times, 17 August 2016

Origins of ARM

"Steve is one of the brightest guys I've ever worked with – brilliant and when we decided to do a microprocessor on our own I made two great decisions – I gave them two things which National, Intel and Motorola had never given their design teams: the first was no money; the second was no people. The only way they could do it was to keep it really simple."

Hermann Hauser, founder of Acorn Computers
(ARM Holdings is a former subsidiary of the now defunct Acorn Computers)

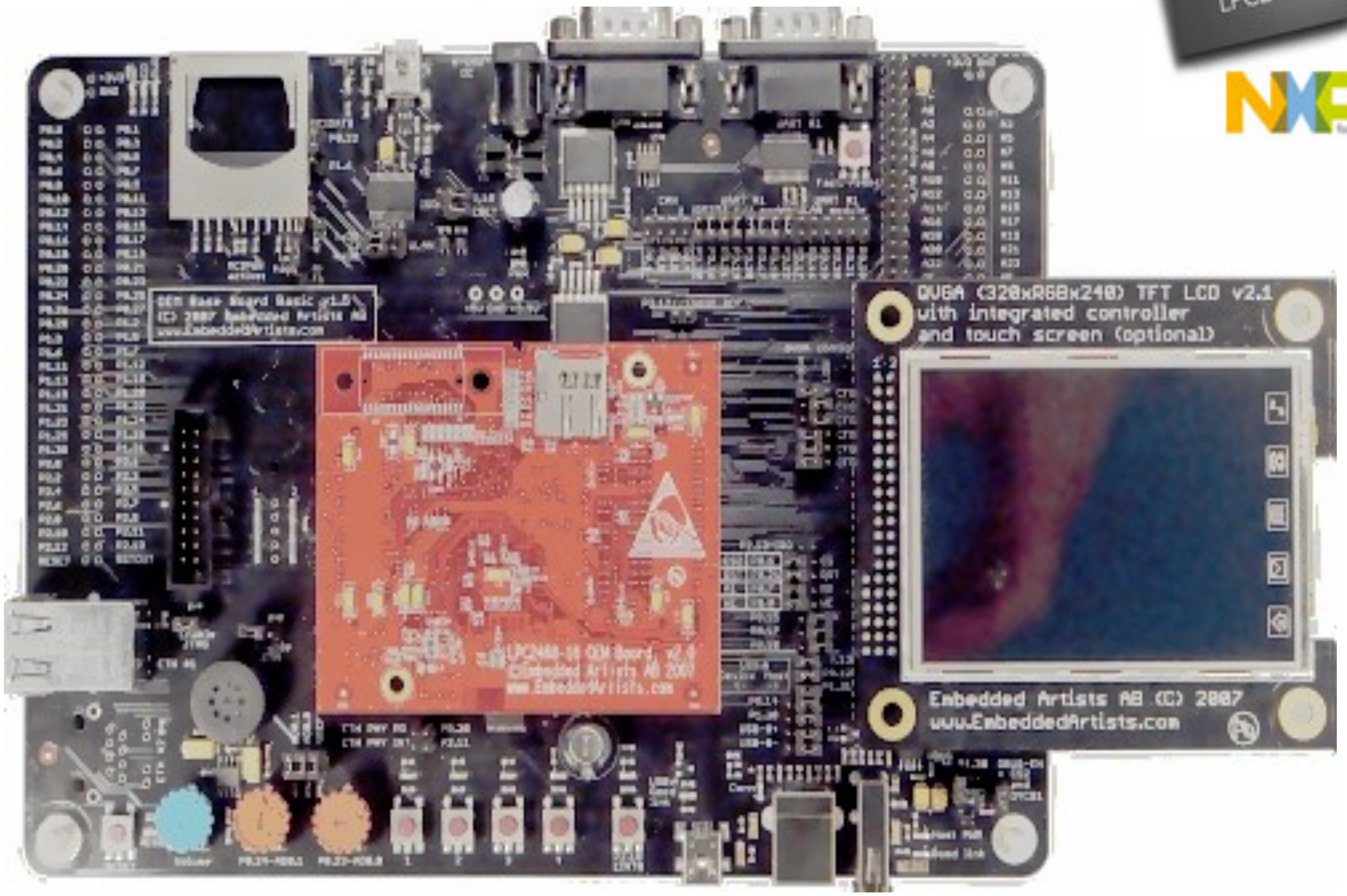
https://youtu.be/Zh96doQ8_as

ARM7TDMI Microprocessor

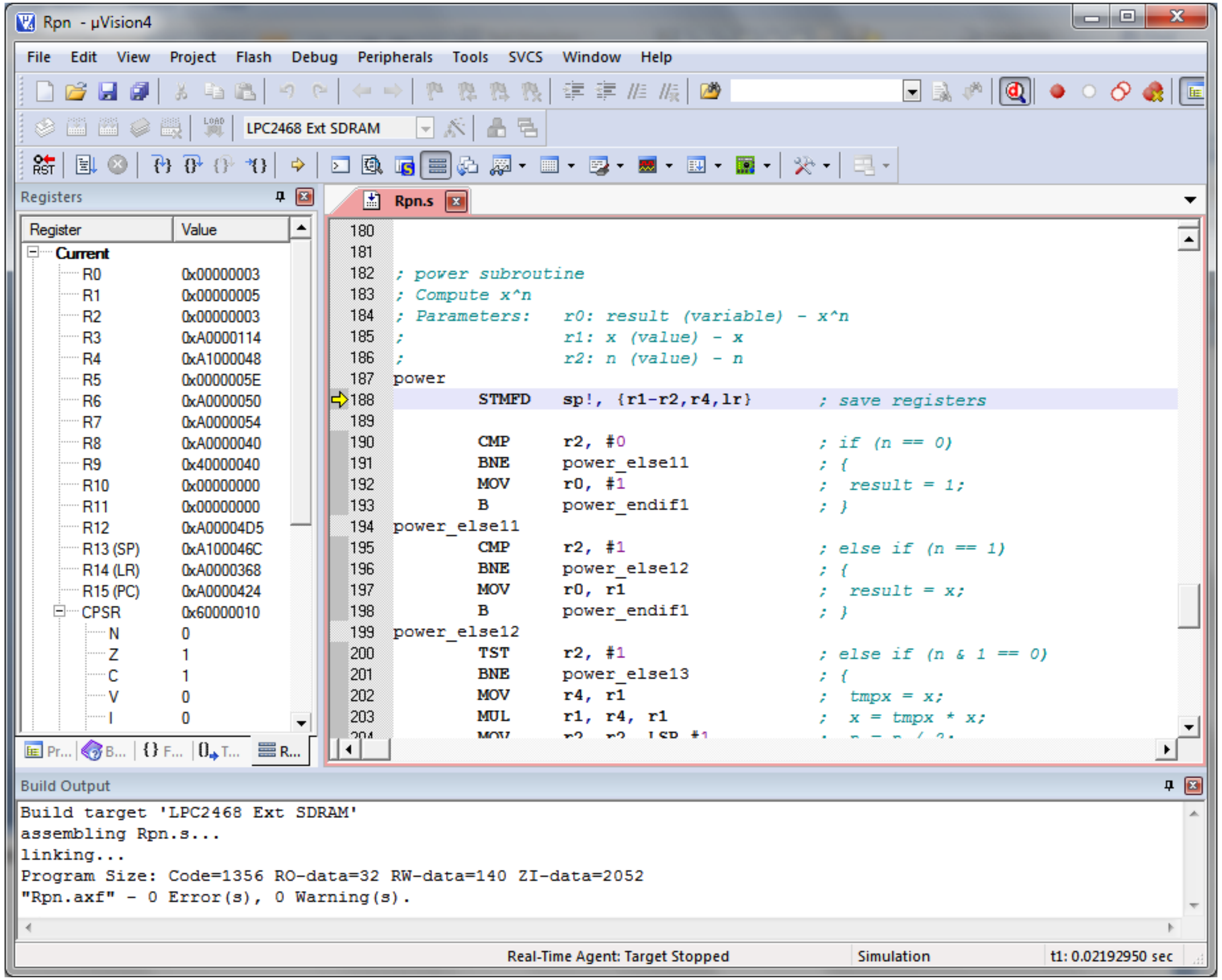
iPod, Nintendo DS, various mobiles, Lego Mindstorms, lots of industrial control applications ...

NXP LPC2468 32-bit **microcontroller**

- ARM7TDMI-S 32-bit **CPU**
- Flash memory (512KiB)
- RAM (96KiB)
- 10/100 Ethernet
- USB 2.0
- A/D & D/A converters
- …



Development Environment



https://www.keil.com/demo/eval/arm.htm

Demonstration

Keil µVision Development Environment

Write a simple program

Assemble the program

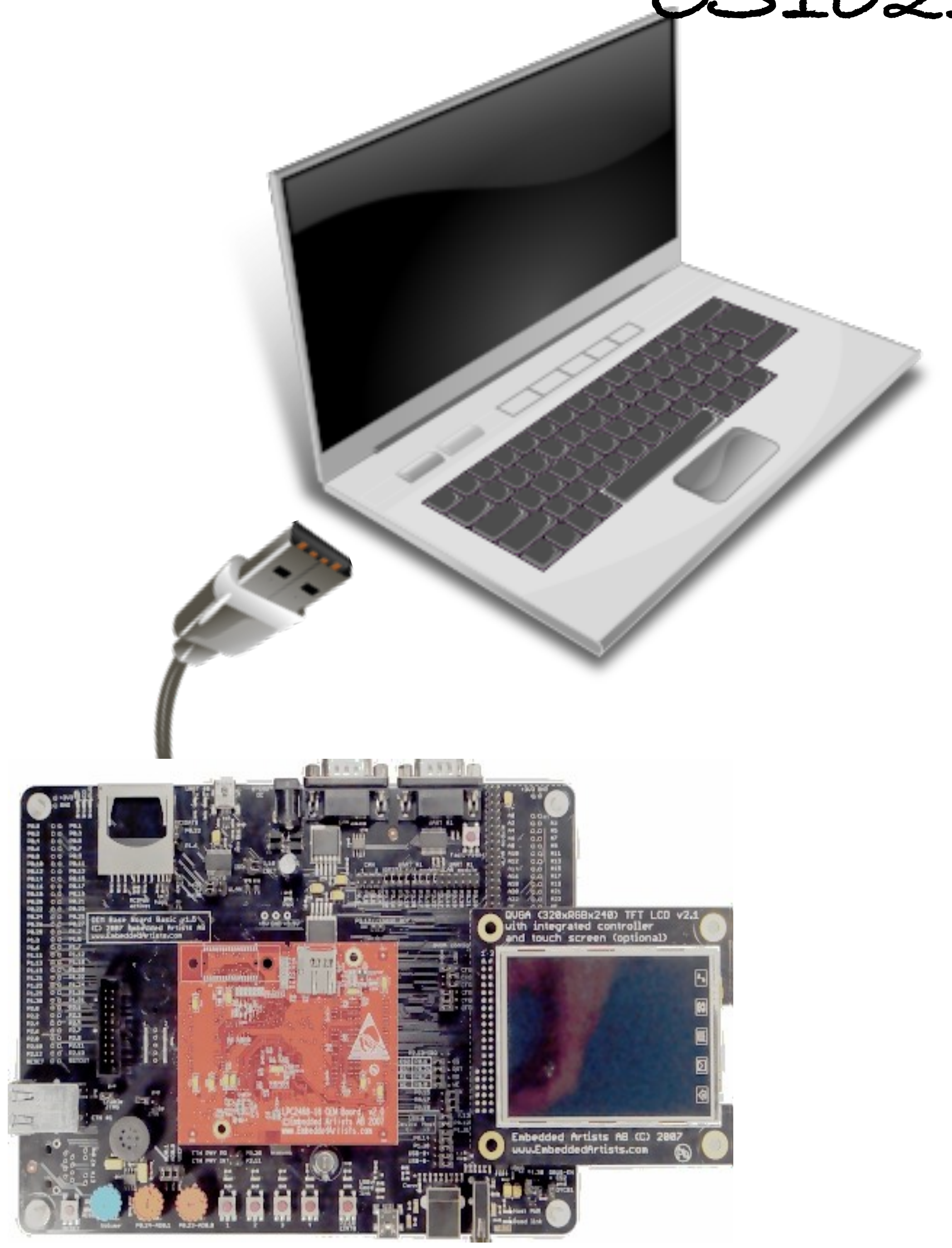
Load the program into memory

Run it and **observe** the results

write program
compile program
run program
observe results

CS1010 Intro to Programming

CS1021 Intro to Computing



write program
assemble program
observe results

load ↓ ↕ debug

run program

Demonstration

A simple program that adds four numbers:

1. Make the first number our total
2. Add the second number to the total
3. Add the third number to the total
4. Add the fourth number to the total

jdukes@tcd.ie

Demonstration

result

first "operand"

second "operand"

```
MOV   total, a               ; Make the first number the subtotal
ADD   total, total, b        ; Add the second number to the subtotal
ADD   total, total, c        ; Add the third number to the subtotal
ADD   total, total, d        ; Add the fourth number to the subtotal
```

a b c d

Call the numbers R1, R2, R3, R4

Call the total R0

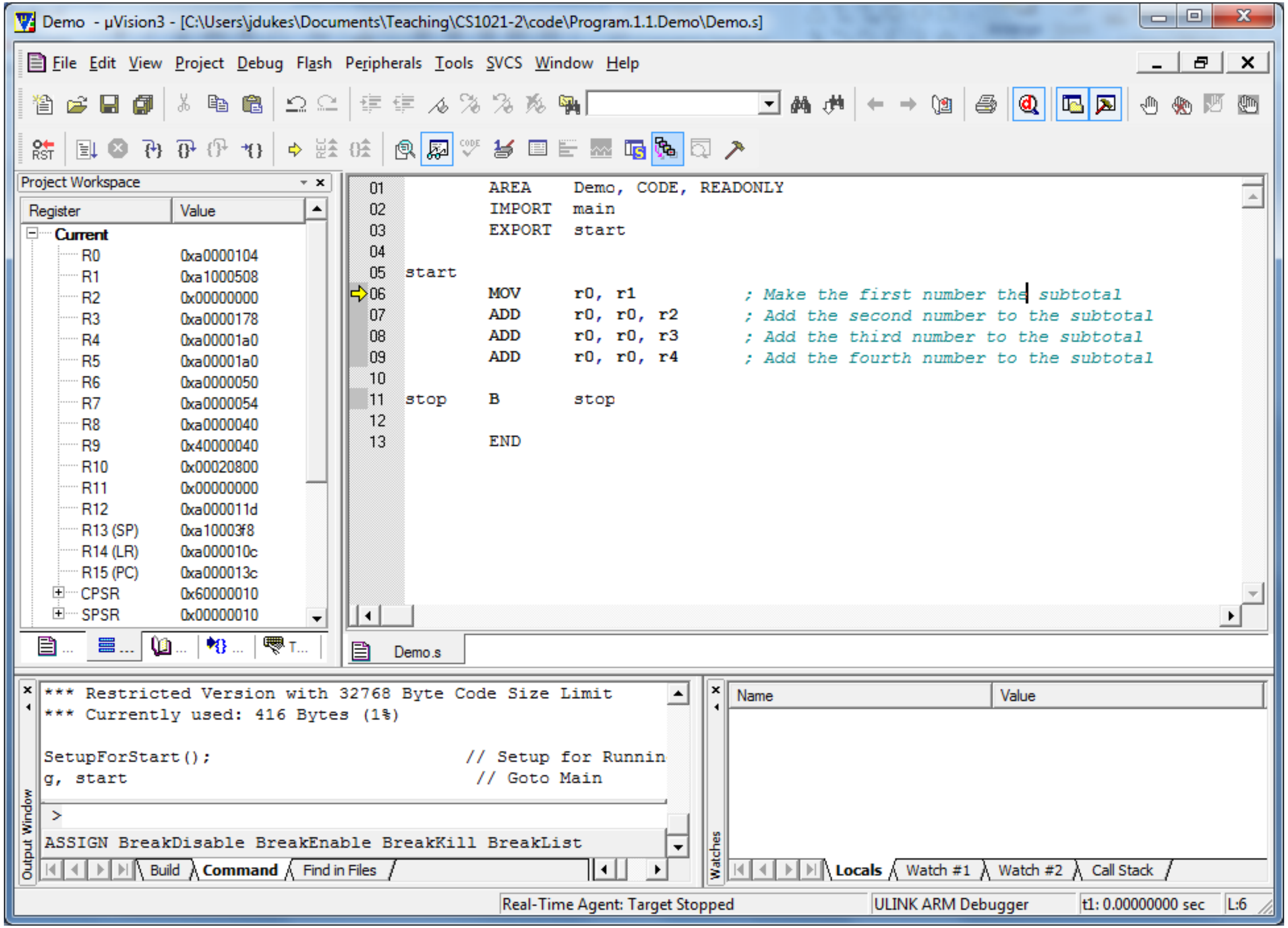
Jonathan Dukes | jdukes@tcd.ie

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
```


Jonathan Dukes | jdukes@tcd.ie

Development Environment



Simple Model of a Microprocessor System

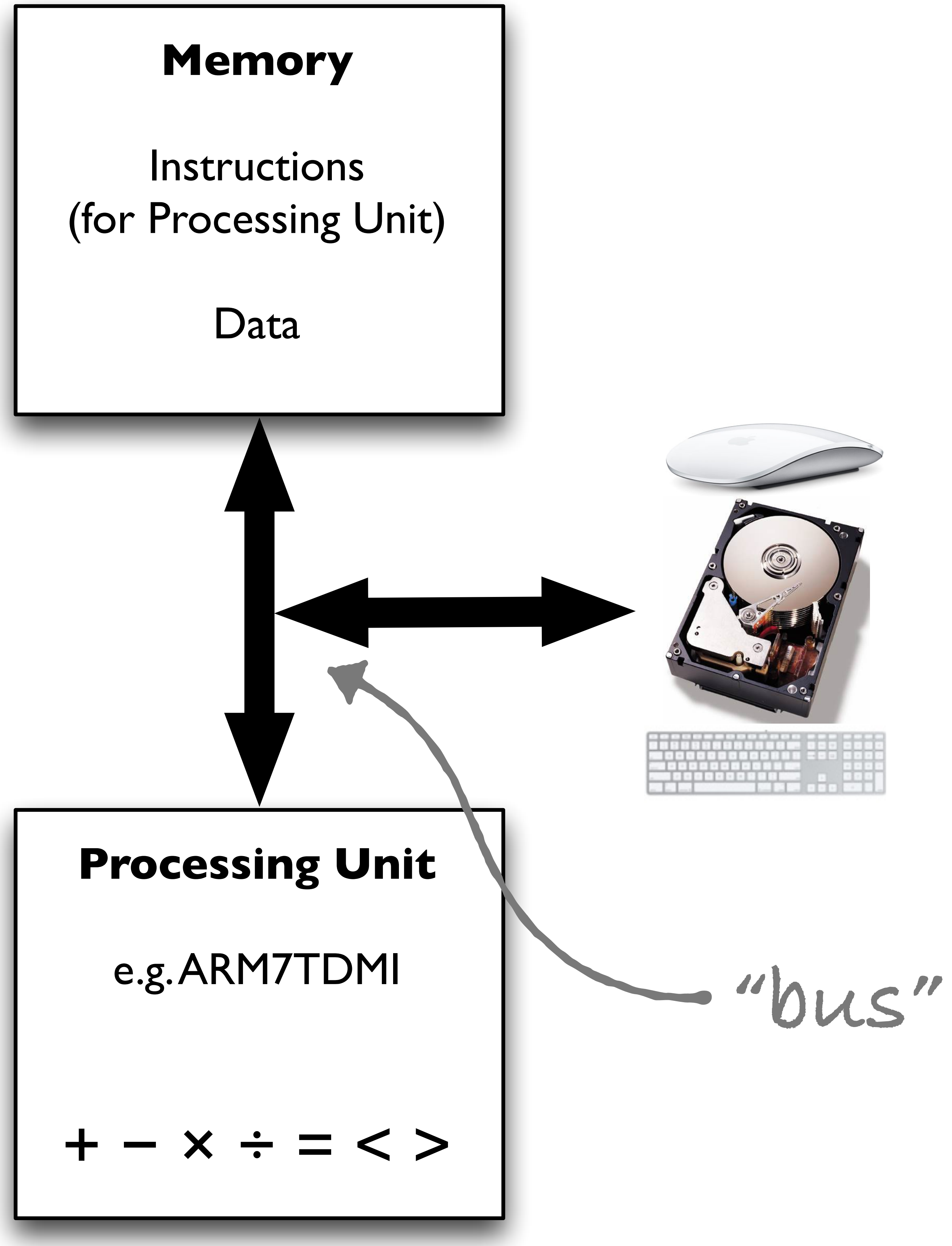
A **processing unit** or **processor** which performs operations on data

Memory, which stores:

- **Data**: representing text, images, videos, sensor readings, π, 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



jdukes@tcd.ie

Simple Model of a Microprocessor System

Memory is arranged as a series of “**locations**”, each of which stores a small piece of information

Each location has a unique “**address**”

The information at each location may be

- **data**, e.g. the value 91 or
- an **instruction** that tells the processor how to manipulate the data

But instructions are also encoded as values!!

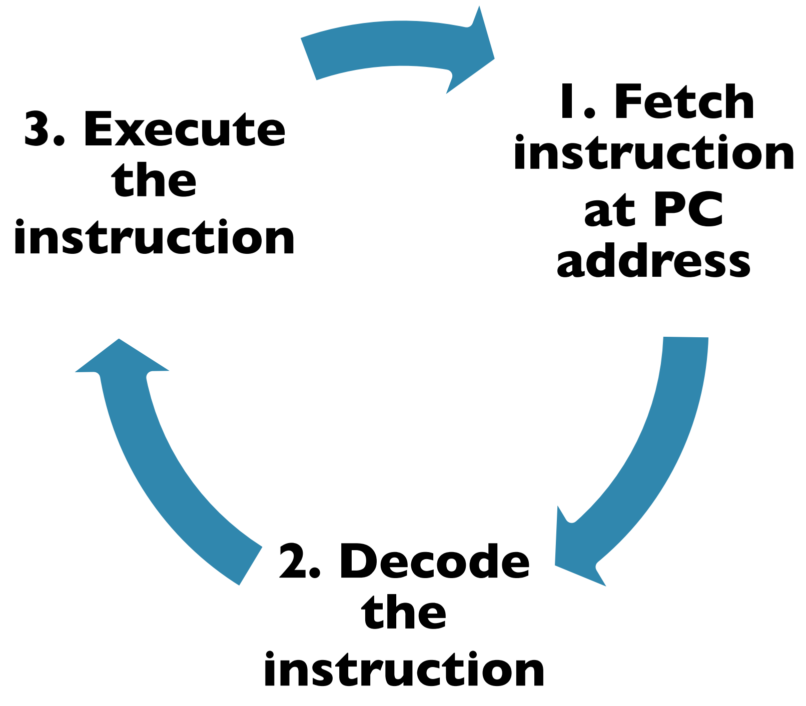
- e.g. the value 91 might be a code used to tell the processing unit to add two values together

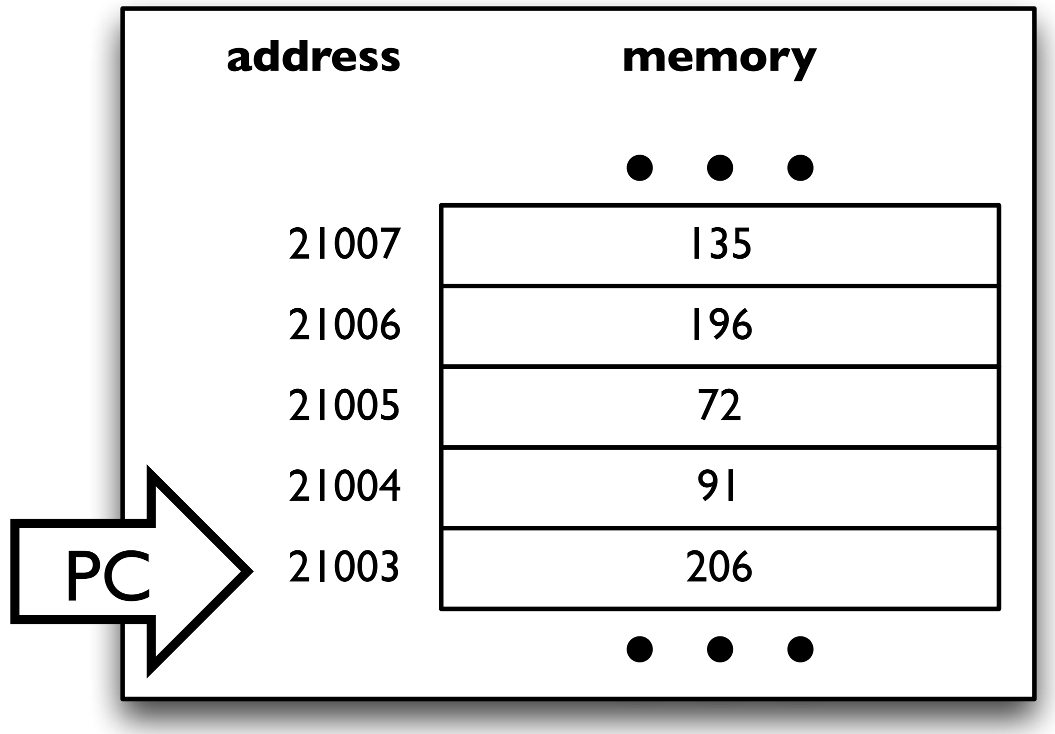
address	memory
	• • •
21013	64
21012	78
21011	251
21010	35
21009	27
21008	89
21007	135
21006	196
21005	72
21004	91
21003	206
21002	131
21001	135
21000	78
20999	109
20998	7
	• • •

jdukes@tcd.ie

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 fetching an instruction, the value of the Program Counter is changed to the address of the next instruction in the program

Processing unit keeps doing this until the computer is turned off

jdukes@tcd.ie

Program Execution

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 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 processor/computer!

jdukes@tcd.ie