Systems and Devices 1
Lec 1: Introduction

Course information

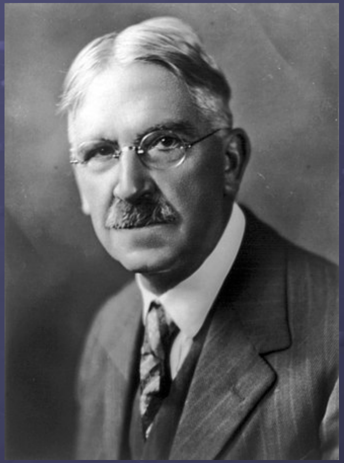
- Lecturer: Michael Freeman
- Office : CSE145, Email : mjf@cs.york.ac.uk
- Home page : http://www.cs.york.ac.uk/~mjf
  - My timetable and other stuff ...
- Teaching Material : http://vle.york.ac.uk
  - Slides, Laboratory scripts, Exercises ...
    - Lecture slides, teaching material containing supporting background material, examples to help illustrate important points.
      - IMPORTANT: you do not need to memorize dates, specific features of different computers ...
- Q&A : if you have any questions do pop in.

Module aims

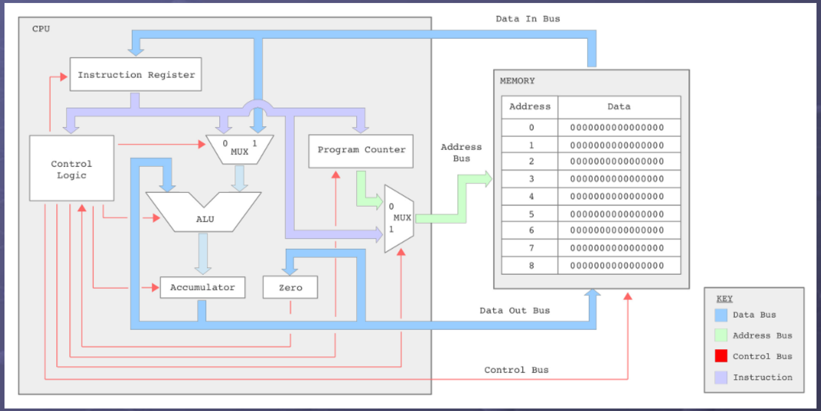
- Module descriptor :
  - https://www.cs.york.ac.uk/modules/
- Students taking this module will gain a:
  - Foundation in the key architectural components of a computer system and how a computer system is constructed.
  - Understanding of how a program implemented in a high level programming language e.g. C executes on that system.
  - Introduced to a bottom-up approach of how a computer system is constructed in hardware, motivated by real examples.

How this module will be taught

- Learning by doing
  - “A theory of education expounded by American philosopher John Dewey. It's a hands-on approach to learning, meaning students must interact with their environment in order to adapt and learn”
  - Therefore, you shall :
    - Build a computer.
    - Write an assembler / compiler.


John Dewey

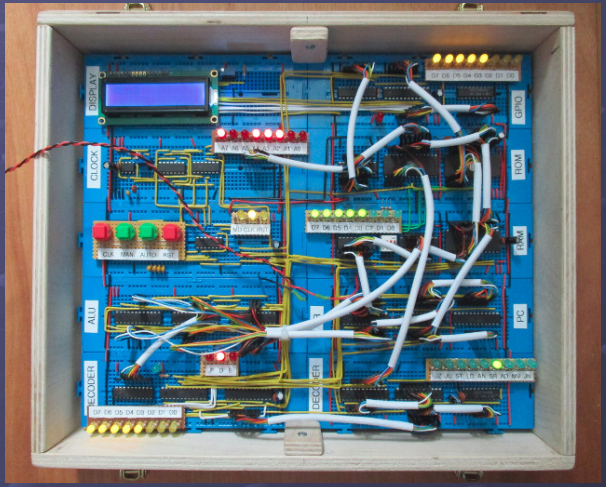
SimpleCPU_v1a



- Block diagram
  - A very simple architecture demonstrating the key architectural components of a computer system

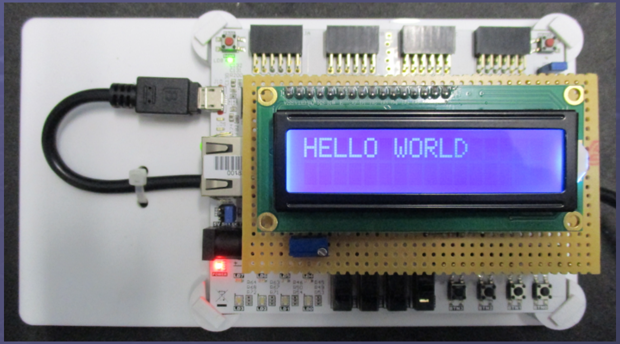
Demo : SimpleCPU_v1a

- Von Neumann architecture
  - 8 bit address bus
  - 16 bit data bus
  - 8 bit accumulator based design
  - 8 bit ALU
    - ADD, SUB, AND
  - 10 instructions
  - 2 addressing modes



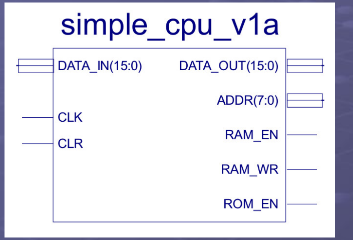
More info : http://www.simplecpudesign.com/

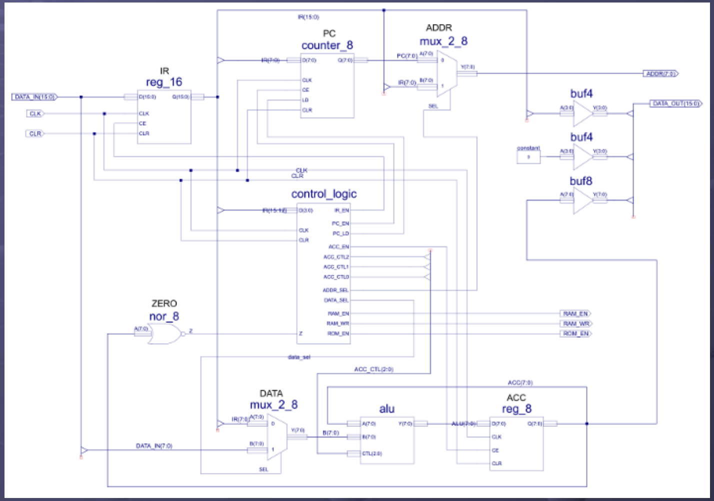
Demo : SimpleCPU_v1a



- However, to save time you will not be “building” the actual hardware, rather you will implement this design in Field Programmable Gate Arrays (FPGA).
  - All components on one device = Faster execution time
    - Minimise communications delays, more operations per second, but ...

SimpleCPU_v1a

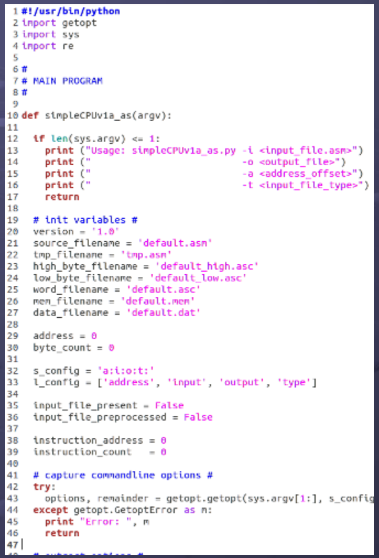




- Processor “built” using Xilinx ISE:
  - Schematic capture.
  - VHDL

Compiler and Assembler

- Implemented in Python
- Assembler
  - A program for converting instructions written in low-level symbolic code into machine code.
    - One-to-one translation, assembler directives, but no optimisation.
- Compiler
  - A program for converting a high level language into low level assembly code.
  - Basic C constructs based on macros
    - IF, FOR, WHILE ...



```
#!/usr/bin/python
import getopt
import sys
import re

#
# MAIN PROGRAM
#

def simpleCPUv1a_as(argv):

  if len(sys.argv) <= 1:
    print ("Usage: simpleCPUv1a_as.py -i <input_file.asm>")
    print ("                          -o <output_file>")
    print ("                          -a <address_offset>")
    print ("                          -t <input_file_type>")
    return

  # init variables #
  version = '1.0'
  source_filename = 'default.asm'
  tmp_filename = 'tmp.asm'
  high_byte_filename = 'default_high.asc'
  low_byte_filename = 'default_low.asc'
  word_filename = 'default.asc'
  mem_filename = 'default.mem'
  data_filename = 'default.dat'

  address = 0
  byte_count = 0

  s_config = 'a:i:o:t:'
  l_config = ['address', 'input', 'output', 'type']

  input_file_present = False
  input_file_preprocessed = False

  instruction_address = 0
  instruction_count   = 0

  # capture commandline options #
  try:
    options, remainder = getopt.getopt(sys.argv[1:], s_config
  except getopt.GetoptError as m:
    print "Error: ", m
    return

```

Timetable

	Spring Term											Summer Term					Total (hours)
Week	1	2	3	4	5	6	7	8	9	10		1	2	3	4		
Lecture		L1 L2	L3	L3	L4	L4	L5	L5	L6	L6							10
Practical		P1	P2	P3	P4	P5	P6	P7	P8	P9		P10	O1	O2	O3		26
Workshop			W1	W2	W3	W4	W5	W6	W7	W8							8
Catch-up			C1		C2		C3		C4								4
																	48

- Pre-reading : 40
- Review material, laboratory and problem class scripts : 40
- Open Assessment : 40
- Reflection, research and revision : 32

Timetable breakdown

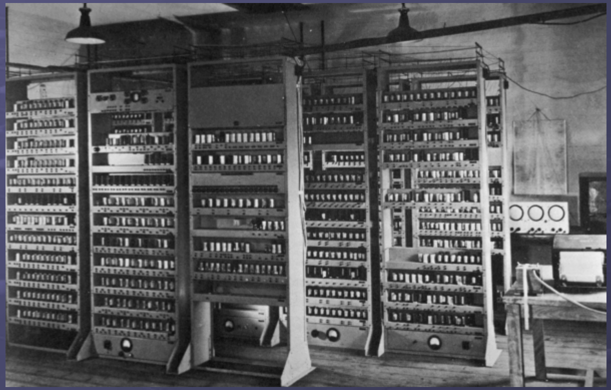
- Lectures and Workshops
  - Lec 01 : Introduction
    - Module aims, Learning outcomes ...
  - Lec 02 : Data types
    - Number bases, Conversion, Boolean logic gates ...
  - Lec 03 : Combinatorial Logic
    - Multiplexers, Encoders, Decoders, Binary arithmetic ...
  - Lec 04 : Sequential Logic
    - Flip-flops, Registers, Counters, Memory ...
  - Lec 05 : The Computer
    - F-D-E, Instruction sets, Assembly language, GPIO …
  - Lec 06 : The CPU

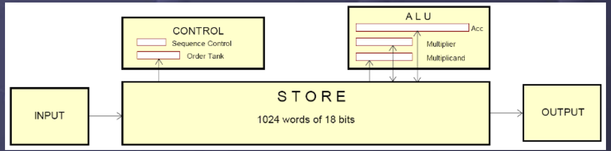
Timetable breakdown

- Practicals
  - Lab 01 : Logic gates and signals
  - Lab 02 : Using logic gates - hardwired controllers
  - Lab 03 : Using logic gates - encoding data
  - Lab 04 : Using logic gates - processing data
  - Lab 05 : Implementing simple state-machines
  - Lab 06 : CPUSim model of SimpleCPU_v1a
  - Lab 07 : SimpleCPU_v1a on an FPGA
  - Lab 08 : SimpleCPU_v1a Hello World
  - Lab 09 : SimpleCPU_v1d and image processing
  - Lab 10 : SimpleCPU_v1d pong

EDSAC

- Electronic Delay Storage Automatic Calculator (1949)
  - The first ‘practical’ computer to execute a stored program.
- From one point of view computer's have not changed.
  - Program's are still defined using instructions stored in memory.

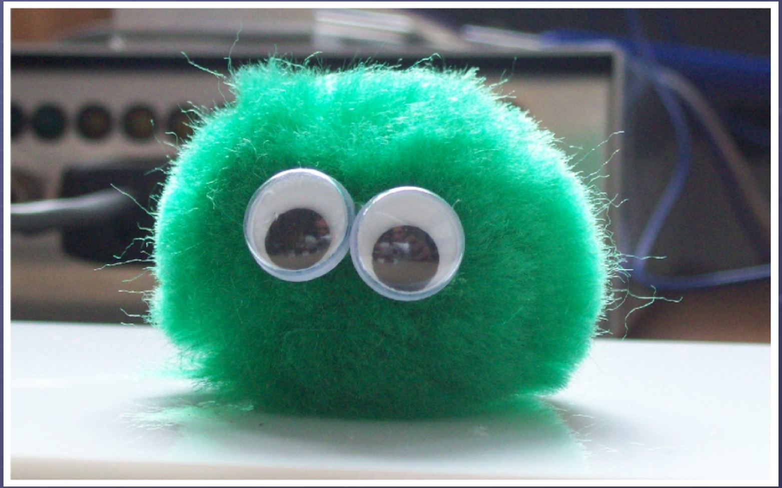




EDSAC

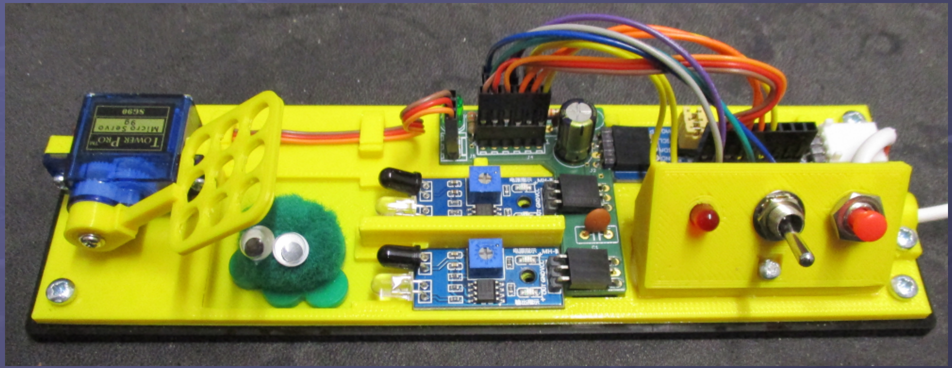
- Another thing in common : the difficulty lies not in writing programs, but in getting them to work.
  - “By June 1949 ... I was trying to get working my first non-trivial program, which was for the numerical integration of Airy’s differential equation. It was on one on my journeys between the EDSAC room and the punching equipment that “hesitating at the angles of the stairs” the realization came over me that a good part of the remainder of my life was going to be spent in finding the errors in my own programs.” M. Wilkes
- Therefore, this module will focus on finding and removing bugs from software and hardware.

What SYS1 is really about



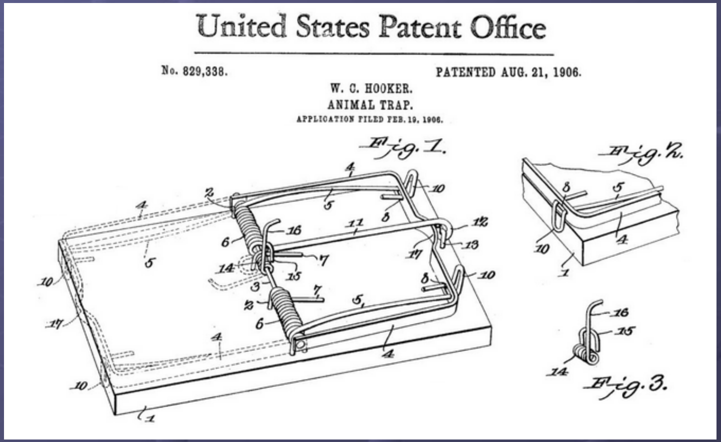
- Finding and removing bugs : Bob

Demo : FPGA Bug Trap



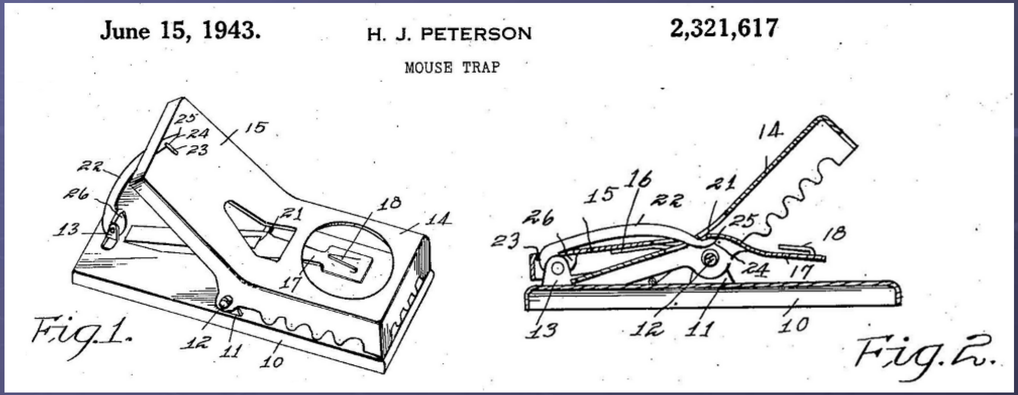
- Bug trap : an embedded system
  - Inputs : Push switch, toggle switch, Infra-red beams front and back
  - Outputs : Servo SWAT, LED

Build a better ...



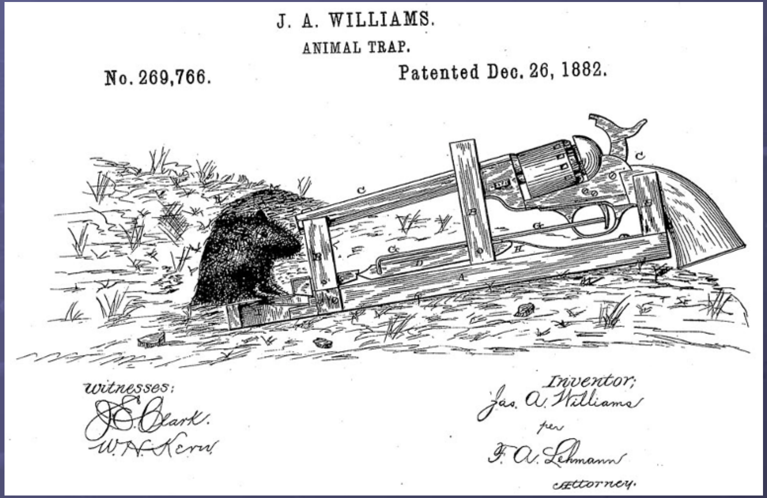
- "Build a better mousetrap, and the world will beat a path to your door" Ralph Waldo Emerson

Build a better ...



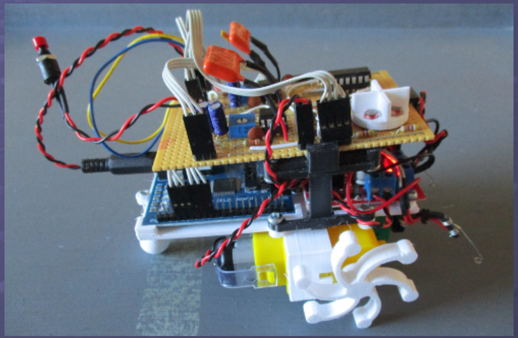
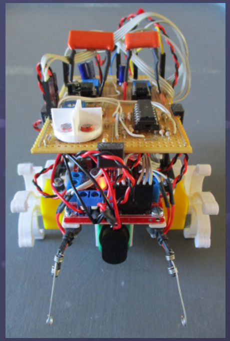
- Designing a better mouse trap is a good analogy for designing a better computer.
  - Identify what is needed, optimise performance, but keep it simple.

Build a better ...



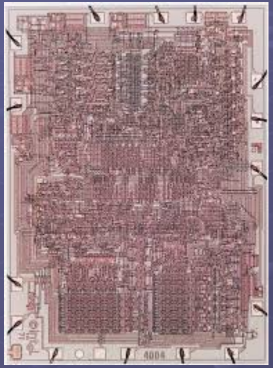
- Not all innovations are a good idea :)

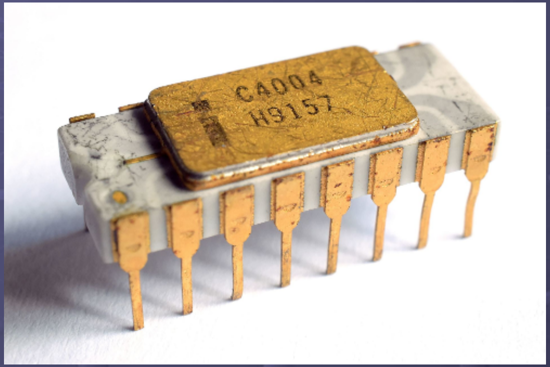
To be fair ...

- Robo-cockroach: summer term week 8 – 10 activity
  - Limited number of free kits, but you can also buy the parts online.
  - Designed as part of weekly workshops.

Intel 4004





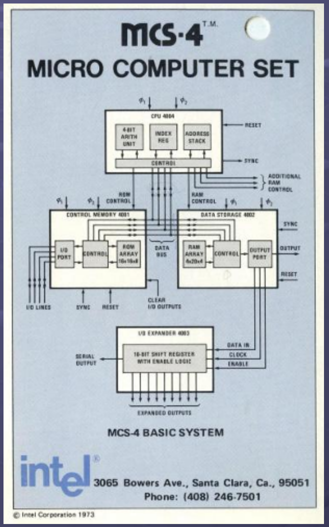
- How do you go about building a better bug-trap?
- Mirror the approach taken by Intel when developing the first commercial CPU : Intel 4004.
  - 4 bit architecture, 2,250 transistors, 740KHz clock

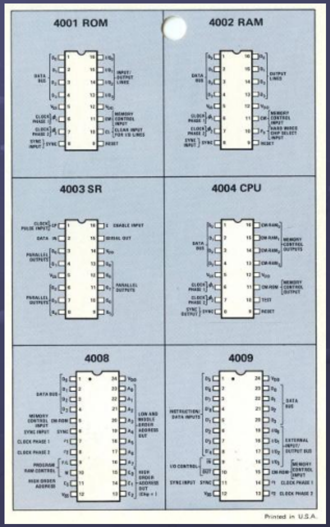
Intel 4004

- Back in the 1970s Intel was a small company developing memory devices.
- Commissioned to produce a set of customs ICs for Busicom Corp.
  - 141-PF electronic calculator
- Development team
  - Marcian Hoff, Federico Faggin and Masatoshi Shima
- Too much work, too little time, too expensive so plan B ...

Intel 4004

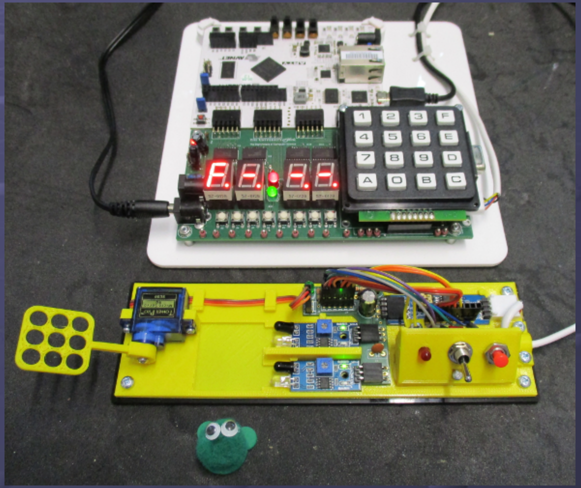
- Solution : develop a general purpose processor
  - Design one set of ICs that can be reconfigured using software to “emulate” different application specific ICs



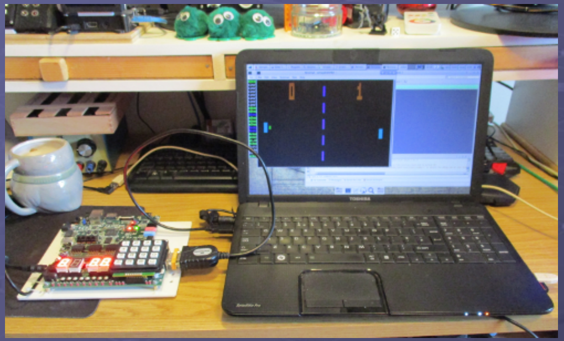


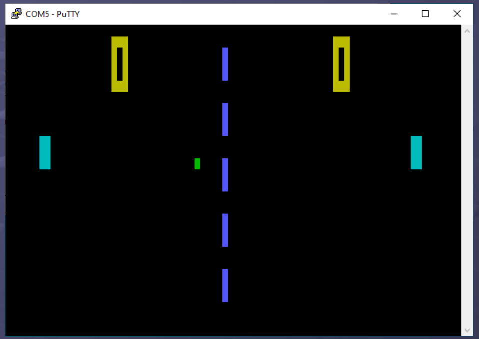
The ultimate bug trap

- Architectures
  - Hard-wired
    - Combinatorial logic
    - Sequential logic
  - Processor
    - Display, Keypad, Real time clock ...
- Some may say a £250 bug trap will never sell. I say you can never have too many buttons and LEDs :)



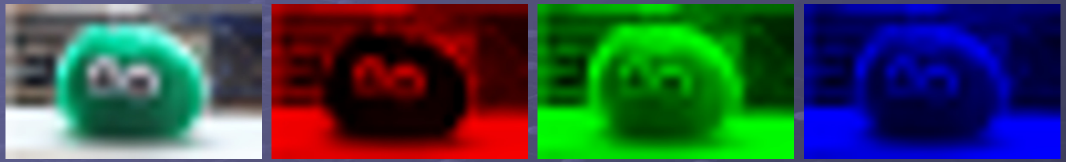
The Bug Game





- Tired of HD graphics, getting eye strain from playing games at 60 fps, then the simpleCPU games machine is for you i.e. design everything, HW & SW.
  - Any similarities to Pong are purely coincidental :)

Open assessment



- Summer term weeks 1 – 4
  - Using the simpleCPU_v1d develop new hardware, instructions and software to implement identified system functions for a camera based bug-trap.
    - SimpleCPU_v1d is an enhanced version of the original architecture with additional instructions and addressing modes targeted at this application domain.
  - Individual assessment, marks based on system performance i.e. the faster the system the higher the mark.

Summary

- At the end of this module a student will be able to:

Learning Outcome	Description
SD101	Identify the purpose of key computer hardware components such as processors, memories and buses.
SD102	Describe different data types commonly found in binary systems (e.g. signed vs. unsigned integers)
SD103	Apply different operations to binary to convert and transform binary strings, perform arithmetic, and perform logical operations.
SD104	Express logical expressions as basic gates, transistors and combinatorial logic circuits
SD105	Describe the function and limitations of a variety of logical building blocks in the context of processor architectures
SD106	Describe the von Neumann Model paradigm of computer architecture, including the fetch execute cycle of instruction processing.

Summary

- At the end of this module a student will be able to:

Learning Outcome	Description
SD107	Explain how operations executed in a processor can be used to implement to higher level sequential, conditional and iterative programming language constructs
SD108	Explain the use of assemblers, compilers and linkers to create executable code for a processor.
SD109	Build a simple system comprised of a CPU, memory and input/output.
SD110	Use a tool-chain (compiling and linking code) to develop software for a simple system built in the module
SD111	Relate the fundamentals of computer architecture to real world systems through exploration of relevant case studies.
SD112	Identify potential security problems associated with architecture design.