Computer Systems Architecture

http://cs.nott.ac.uk/∼txa/g51csa/

Thorsten Altenkirch

School of Computer Science and IT
University of Nottingham

Lecture 02: Basic architecture; Introduction to SPIM

The University of Nottingham

What is computer memory?

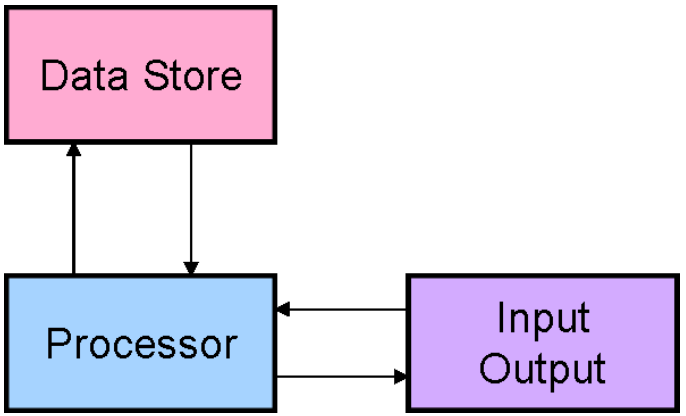
Addr.	Data
0000	'H'
0001	'e'
0002	'l'
0003	'l'
0004	'o'
0005	'_'
0006	'W'
0007	'o'
0008	'r'
0009	'l'
000*A*	'd'
...	...

- Storage device for a computer
- Sequence of data locations
- One unit of data at each 'address'
- Read-Only Memory (ROM)
- Random-Access Memory (RAM)
  - Writable, but volatile!
- EEPROM, Flash etc.

What is data?

- Numbers – binary encoding
- Characters – ASCII, Unicode
- Strings – sequences of characters
- Audio
  - 1-D array (time) of sound pressure
- Images
  - 2-D array (X-Y coordinate) of colour intensities
- What else? ...
- Programs!

What is a computer?



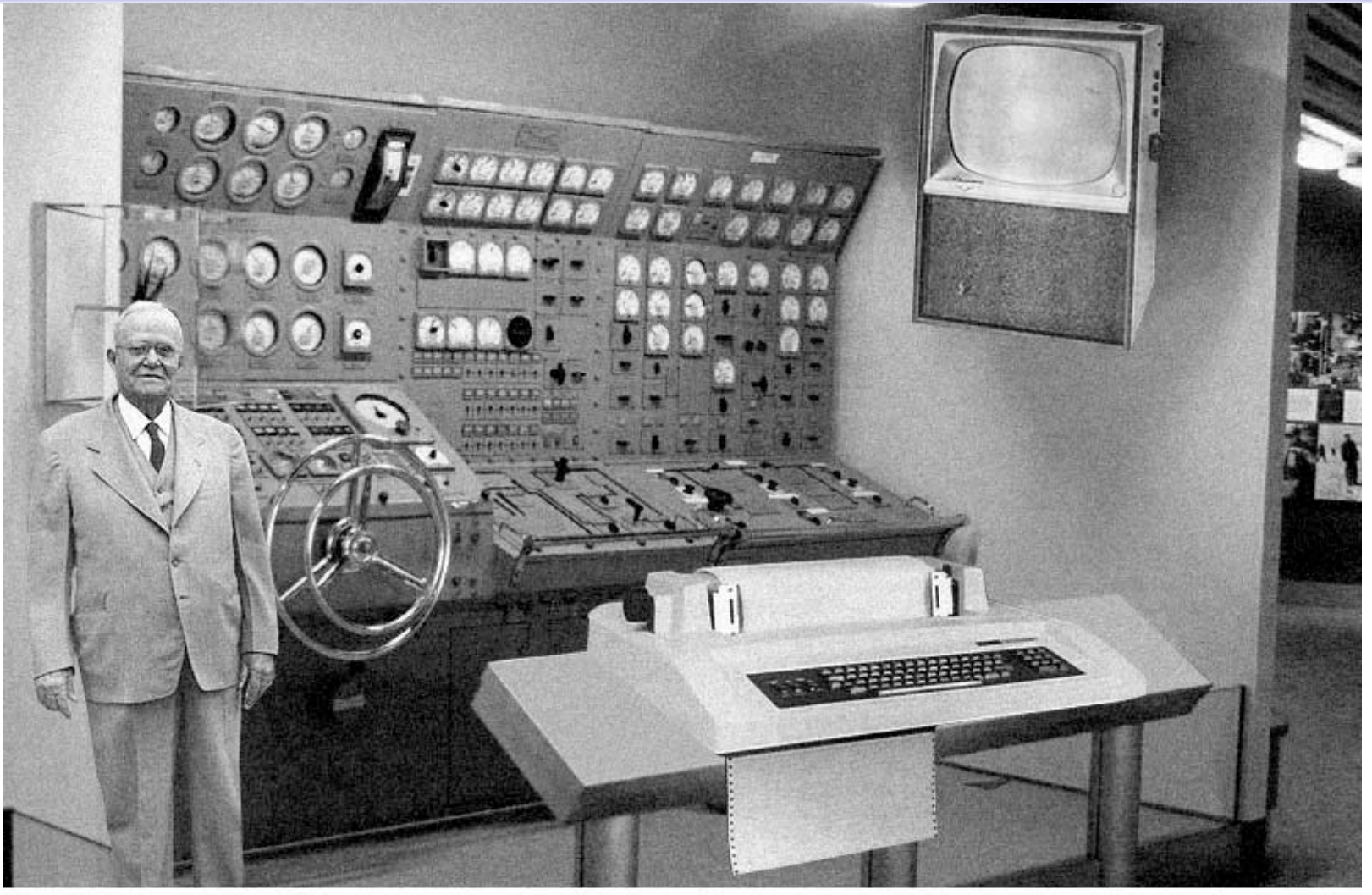
Characteristics

- Recall information
- Process data
- Handle input and output

Are these computers?

- Pocket calculator?
- Digital watch?

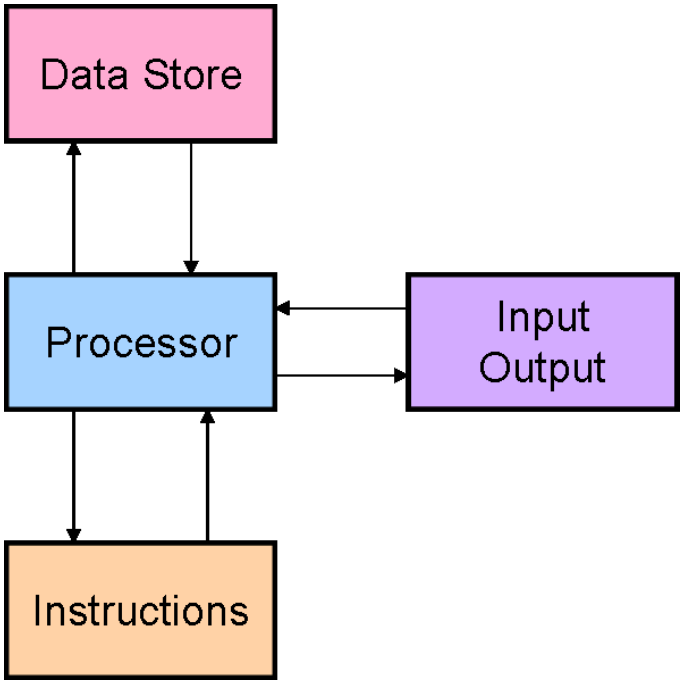
Early computers



Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 50 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use and only

The Harvard architecture

- Program decoupled from the processor
- A different program can be loaded each time

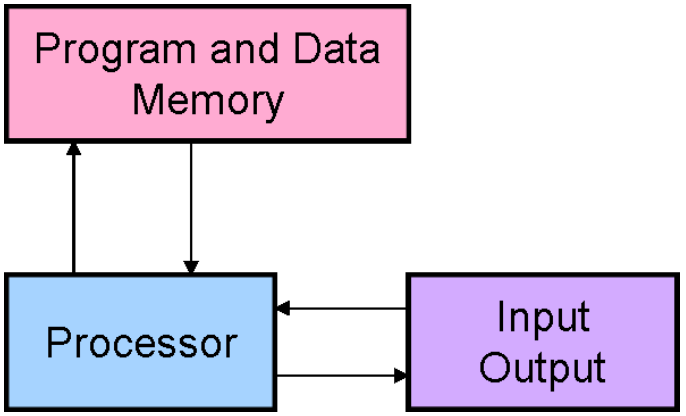


- Small set of instructions
  - Many different programs
- One of the earliest designs
- Now still used in some embedded processors
  - Digital Signal Processors
  - Microcontrollers

The University of Nottingham

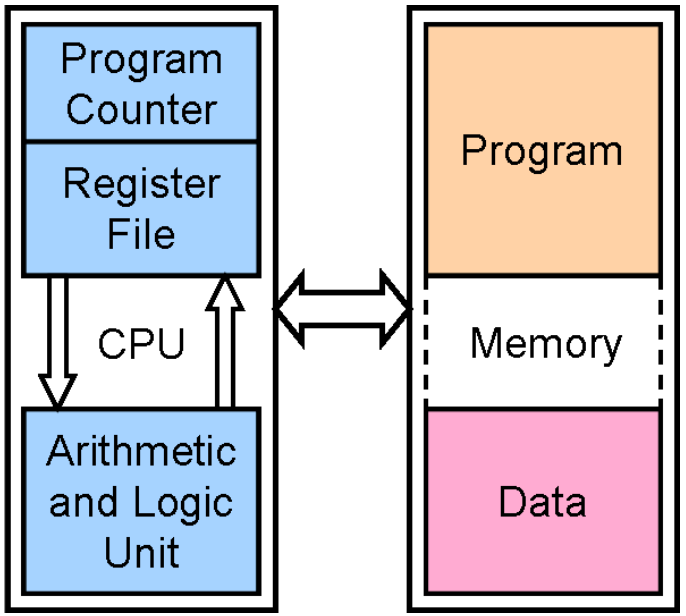
The von Neumann architecture

- But instructions (i.e. programs) are data!
- Attributed to John von Neumann (1903–1957)
- Memory shared between program and data



- Programs can process other programs as input or output
  - Compilers
  - Debuggers
  - Operating systems
- Includes most modern processors

Operation of a von Neumann processor



The fetch-execute cycle

- Fetch (PC and Control)
  - Fetch word at (PC)
  - Instruction decoded
  - PC incremented
- Execute (ALU and Control)
  - Read operands
  - ALU executes operation
  - Result written back

Exercise: run this program, on paper

- Consider a hypothetical 32-bit processor
  - Instructions in English here – no need to 'decode'
- Two general purpose registers, `r0` and `r1`
- The special `pc` register is initially 0000

Address	Instruction / Data
0000	load the word at address 0018 into `r0`
0004	load the word at address 001C into `r1`
0008	put the sum of `r0` and `r1` into `r0`
000C	put the sum of `r0` and `r1` into `r0`
0010	store the word in `r0` into address 0020
0014	stop
0018	00000001_{16}
001C	00000005_{16}
0020	$0000002A_{16}$

The University of Nottingham

Instruction Set Architecture (ISA)

- Specification of processor's basic operations
  - Usually named after the processor they come from
    e.g. 80386, PowerPC, IA-64, MIPS32, ARM
- Instructions are building blocks for programs
- Processors with compatible ISAs can run the same code
  - Some AMD processors can run Intel 80x86 programs
  - Manufacturers like to define their own proprietary ISAs!
  - But they do try to ensure backwards compatibility

The MIPS architecture

- Microprocessor without Interlocked Pipeline Stages
- Designed in early 1980s; team lead by John L Hennessey
- Powered SGI workstations until mid-1990s
- Still widely used in embedded markets
  - PlayStation and PS2, handheld computers
  - Network hubs/switches, WiFi access points
- Typical RISC (Reduced Instruction Set Computer) design
  - Keep instructions simple and fast; use pipelining
- Contrast with a CISC (Complex Instruction...) design
  - Expensive/slow memory; writing machine code difficult
  - So each instruction does as much as possible
  - e.g. `AAA` on 80x86

The University of Nottingham

Assembly programming?

Machine Code

```
01284020
```

English

"put sum of `$t0` and `$t1` into `$t0`"

- Assembly language is a compromise
  - Programmers don't like machine code
  - Computers can't understand English
- Mnemonics – symbolic names for instructions
- Labels – named memory locations

Instruction	Description
`ld $t0, (foo)`	load the word at address *foo* into `$t0`
`add $t2, $t1, $t0`	put the sum of `$t0` and `$t1` into `$t2`
`st $t0, (bar)`	store the word in `$t0` into address *bar*

The University of Nottingham

"Hello World" in MIPS assembly

```
        .data
hello:
        .asciiz "Hello World!\n"
        .text
        .globl main
main:
        la $a0, hello
        li $v0, 4  # print_string
        syscall
        li $v0, 10  # exit
        syscall
```

Assembly syntax

- Assembler *directives* begin with a '.', for example:

  `.data` start assembling data used by the program

  `.asciiz` places a *NUL*-terminated string in memory

  `.text` start assembling program instructions

- *Labels* are names followed by a colon ':'
  - Descriptive names chosen by the programmer
  - The program starts at the `main` label
- *Comments* begin with a '#', until end of line
- Remaining *non-empty* lines are instructions
- Reference manual: Hennessey and Patterson, Appendix A

Installing SPIM

- 'Start'→'Settings'→'Control Panel'
  →'Add or Remove Programs'→'Add New Programs'
  →'PCSpim'→'Add'
  - You should only need to do this once
- Read "Getting Started with PCSpim"
  http://www.cs.wisc.edu/~larus/PCSpim.pdf
- Hennessey and Patterson, Appendix A
  http://www.cs.wisc.edu/~larus/HP_AppA.pdf
  - Comprehensive *reference manual* for SPIM and MIPS-32

Using SPIM; running "Hello World"

- Type in the "Hello World" listing using Notepad
  - Save the file as '`ex1.s`'
- Start SPIM, 'File'→'Open...', and locate above file
- If it asks "Clear program...?", say 'Yes'
- 'Simulator'→'Go' (or `F5`) starts the program
- Other useful commands
  - 'Simulator'→'Single Step' (or `F10`)
  - 'Simulator'→'Reload...'

Thursday quiz

Most of the following questions are multiple choice. There is at least one correct choice but there may be several. For each of the questions list all the roman numerals corresponding to correct answers but none of the incorrect ones.
Questions are marked as follows:

no errors	5 points
1 error	3 points
2 errors	1 point
$\geq$**3 errors**	0 points

Thursday quiz

1. Which of the following boolean expressions is equivalent to $A \wedge \neg B$?
   - a $\neg(A \vee \neg B)$
   - b $\neg A \vee B$
   - c $\neg A \vee \neg\neg B$
   - d $\neg(\neg A \vee B)$
   - e $\neg(\neg A \vee \neg\neg B)$

Thursday quiz

2. Translate 102_{10} into binary and hexadecimal.
   5 points, if both are correct, 3 points if only one is correct.

Thursday quiz

3. Which of the following statements about 8 bit binary numbers is correct?
   a. The largest representable number is 256.
   b. If the number is even, the last digit is 0.
   c. Dividing by 2 corresponds to shifting the number to the right.
   d. The most significant digit has the weight 128.
   e. We do not write leading zeroes.

Thursday quiz

4. Which of the following statements about ASCII are correct?
   a. ASCII stands for *American Standard Computer International Interface*
   b. ASCII defines 256 codes, it requires 8 bits.
   c. ASCII includes control codes for mechnical teleypes.
   d. ASCII inludes codes for Umlauts (e.g. ä) and accented characters (e.g. á).
   e. In ASCII all characters are always capitalized.

Thursday quiz

5. What is a word?
   - a A word is always 4 bytes.
   - b A word is the amount of data a processor can process in one step.
   - c A word is a sequence of characters stored in computer memory.
   - d On a 64-bit processor a word is 8 bytes.
   - e In a word the least significant byte always comes first.