Overview: Computer Architecture

- motivation
- approach
- Instruction Set Architecture
- RISC and CISC
- MIPS architecture and instructions

  - 18 lectures, 8 tutorials

  - homepage:
    https://www.doc.ic.ac.uk/~wl/teachlocal/arch2

Why study architecture? The Good..

- understand:
  - how computers work; bridge hardware/software gap
  - choices and constraints for computer engineers/architects
  - how to manage complexity of architecture description/design
- undergoing rapid development
  - applications: internet, medical imaging, cloud computing
  - technology: build your own reconfigurable processor
    customisable parallelism: trading speed, power, accuracy...
  - technology: non-graphics programs on graphics processor

  new super-computer!
- it has impact on almost all aspects of computing and engineering, on both theory and practice
- example: accelerator architectures for data centre and IoT

Accelerate clouds: Microsoft + Amazon

www.top500.org/news/microsoft-goes-all-in-for-fpgas-to-build-out-cloud-based-ai/

Microsoft Goes All in for FPGAs to Build Out AI Cloud

Michael Feldman | **September 27, 2016 08:42 CEST**

Software giant bets the (server) farm on reconfigurable computing

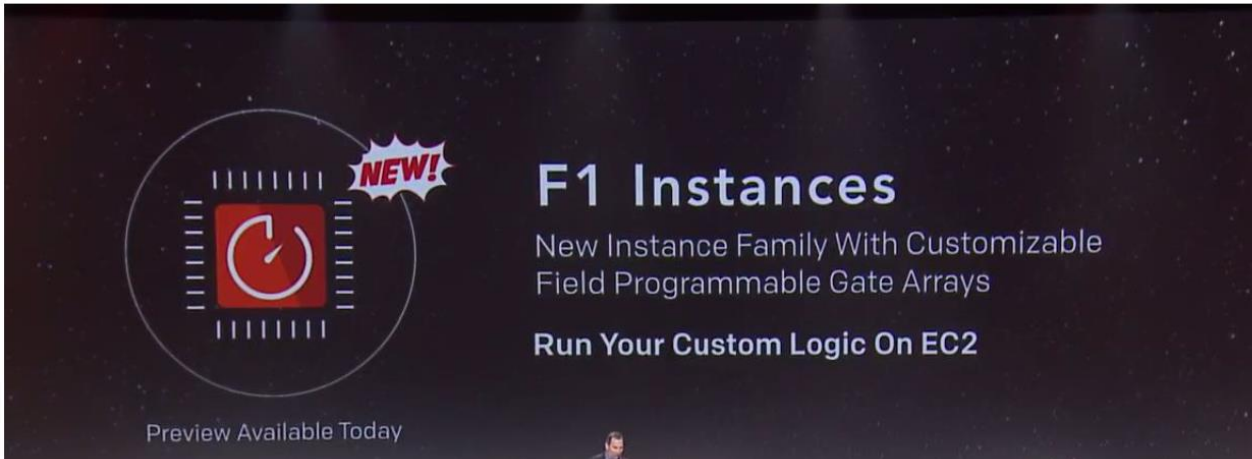
Microsoft has revealed that Altera FPGAs have been installed across every Azure cloud server, creating what the company is calling "the world's first AI supercomputer." The deployment spans 15 countries and represents an aggrega
announcement was made by Microsoft CEO S
the opening keynote at the Ignite Conference

Amazon EC2 F1 Instances

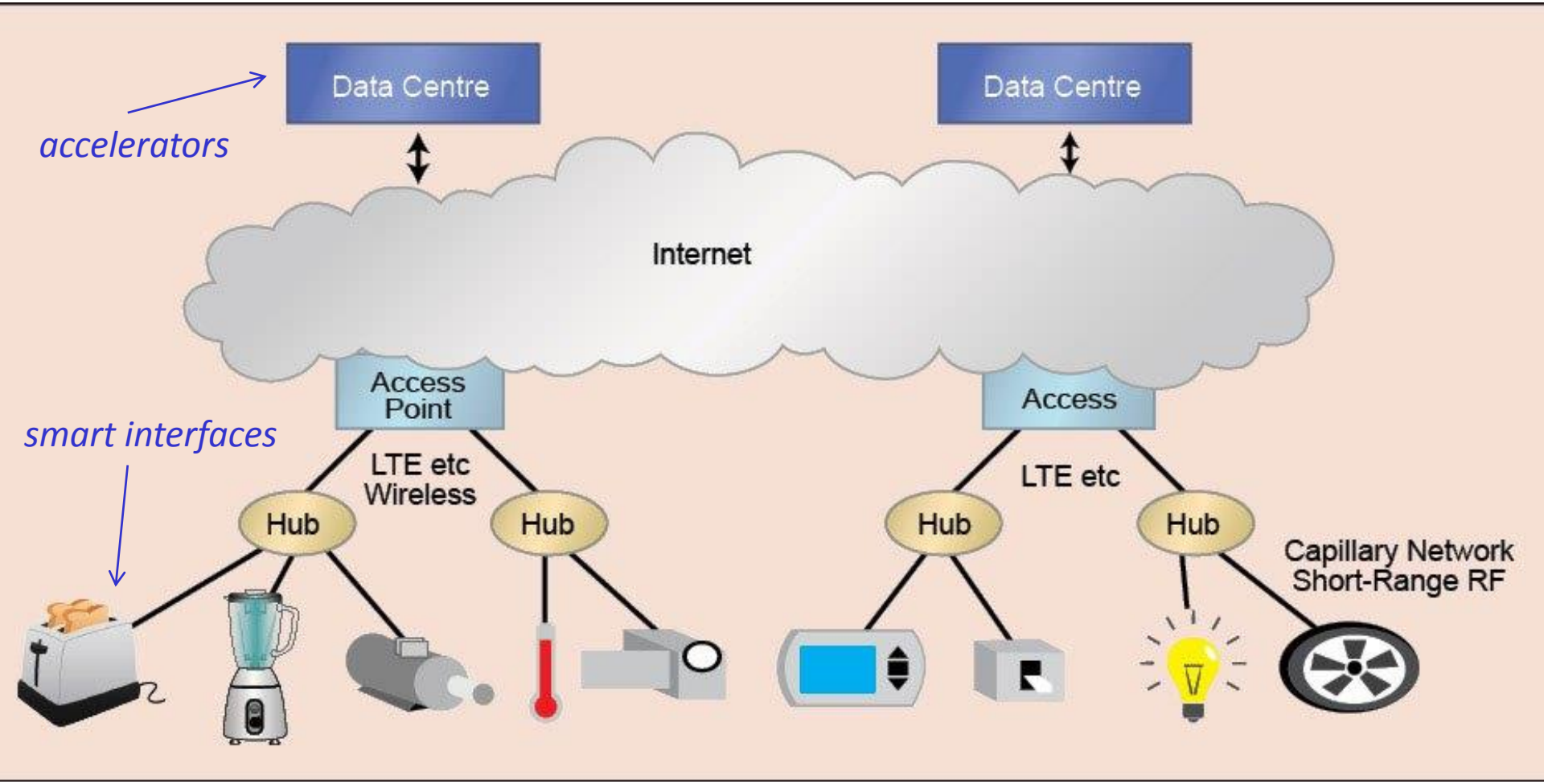
Run Custom FPGAs in the AWS Cloud

Amazon EC2 F1 is a compute instance with field programmable gate arrays (FPGAs) that you can program to create custom hardware accelerations for your application.

aws.amazon.com/ec2/instance-types/f1/

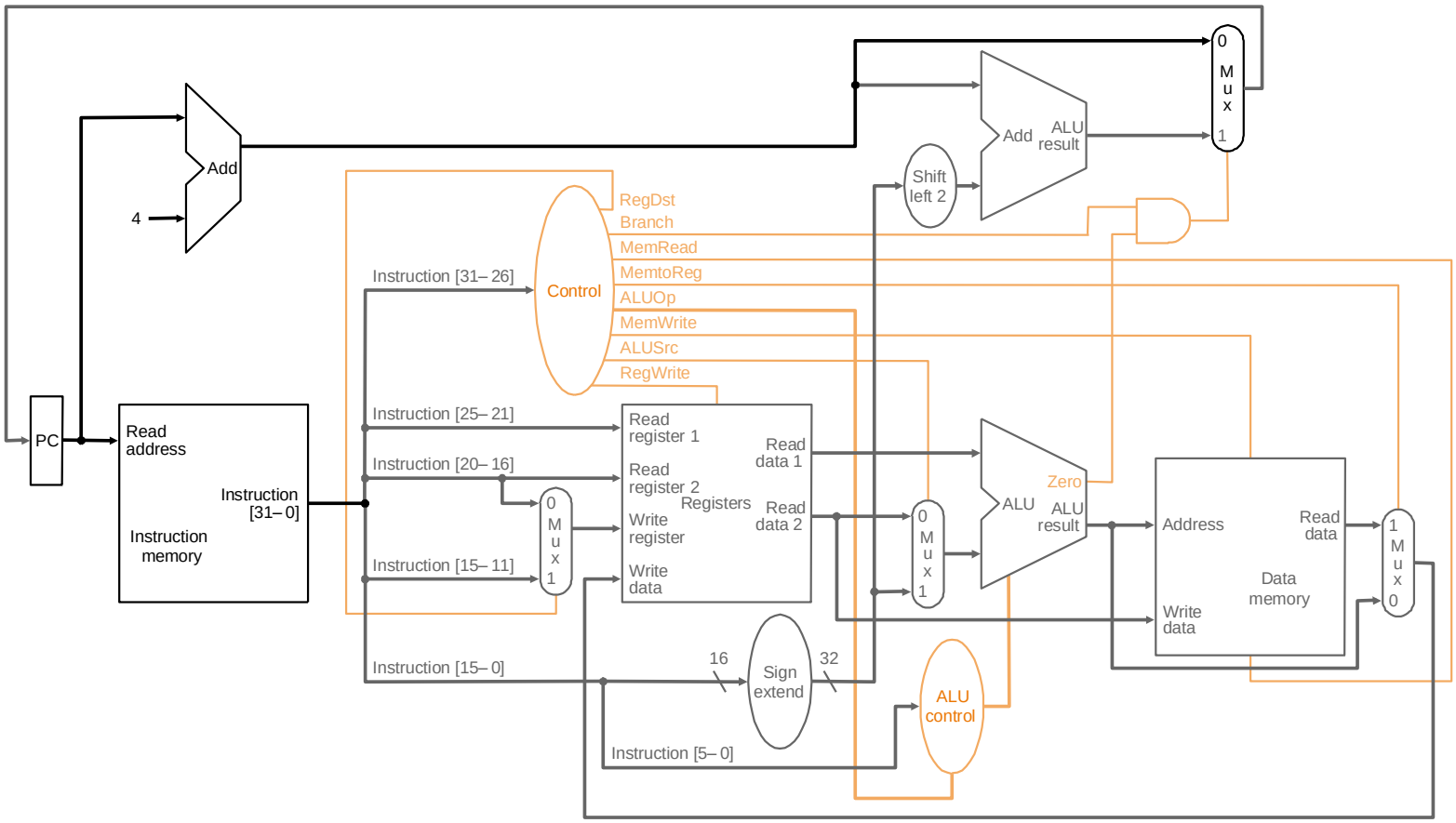


Internet of things



Source: http://electronicsofthings.com

…the Bad and the Ugly



…and this is not the worst!

Hints for success

- come to lectures
- come to tutorials and ask questions
- read notes and course textbook
- attempt unassessed coursework <u>without</u> reading the solutions
- discuss regularly with a friend
- explain ideas to non-specialists
- do industrial placement
- do a research project

Summer research opportunities

- ideal for:
  - gaining experience and a taste of leading-edge research: what university teachers do when we are not teaching
  - preparing final-year project or industrial placement: helped student win Best UK computing project!
- my projects
  - may involve high-tech firms: e.g. Altera, Maxeler
  - flexible start/end dates, around 10-12 weeks
  - UROP bursary available, overseas students OK
  - see www.doc.ic.ac.uk/~wl/summer.html
- other lecturers also have projects

Accelerating Sequential Monte Carlo Method for Real-time Air Traffic Management

summer project students

Thomas C.P. Chau[1], James Targett[1], Marlon Wijeyasinghe[2], Wayne Luk[1], Peter Y.K. Cheung[2], Benjamin Cope[3], Alison Eele[4], Jan Maciejowski[4]

[1] *Department of Computing,* [2] *Department of Electrical and Electronic Engineering*
Imperial College London, United Kingdom
[3] *Altera Europe Limited, United Kingdom*
[4] *Department of Engineering, University of Cambridge, United Kingdom*
email: {c.chau10, james.targett10, marlon.wijeyasinghe09, w.luk, p.cheung}@imperial.ac.uk,
bcope@altera.com, {aje46, jmm1}@cam.ac.uk

ABSTRACT

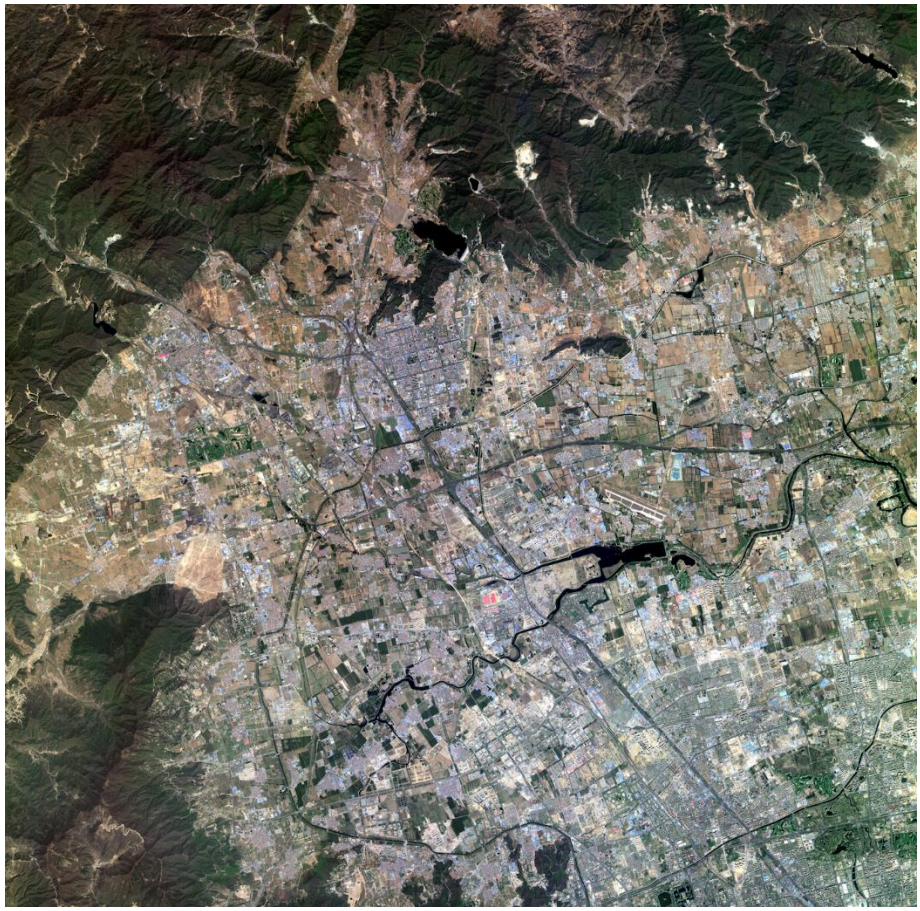
This paper presents how field-programmable gate arrays (FPGAs) are used to accelerate the Sequential Monte Carlo method for air traffic management. A novel data structure is introduced for a particle stream that enables efficient evaluation of constraints and weights. A parallel implementation for this streaming data structure is designed, and an analytical model is provided for estimating the performance and resource usage of our implementation. We compare our design to implementations on CPU and GPU. We show 9.3 times speed up and 89 times improvement in energy efficiency over a Intel X5650 CPU with 12 threads and demonstrate 1.3 times speed up and 13.5 times improvement in energy efficiency over an NVIDIA Tesla C2070 GPU with 448 cores. We also estimate the performance of FPGA in future scenario and show that FPGA is able to control 15 times and 2.8 times more aircraft than CPU and GPU in real-time respectively.

of safe separation between aircraft [2]. Nowadays the pro-
cess is largely performed by humans but attempts are be-
ing made to automate this process [3]. The current ATM
system is near the upper limit of traffic it can safely ac-
commodate [4]. The level of anticipated growth in avia-
tion travel is predicted to double in the next 20 years [5, 6]
SMC has been studied extensively in controlling air traf-
fic [2, 7]. In [2], SMC is applied on scenarios with multiple
aircraft flying under the effects of wind and additional un-
certainty. It is observed that the time required to solve such
problems is currently prohibitive, not to mention meeting
future requirements [2]. Various attempts have been made
to simultaneously process SMC using multi-threaded CPU
or GPU, in applications such as object tracking [8], signa
processing [9] and robot localisation [10]. However, due to
complicated iterative nature of ATM, research in accelerat-
ing SMC for ATM through parallelisation is limited. The
sequential implementation of ATM does not have sufficient
speed for real-time practice [2].

Real-time deep learning: robots + satellites

Facilities inspection

Land use analysis



Approach

- learn ***Computer Organisation and Design*** based on Patterson & Hennessy (3rd, 4th or 5th edition)

  Morgan Kaufmann 2005, 2009, 2014

  - 3rd Ed: chapters 2 to 7, appendices B, C + other material
  - 4th Ed: chapters 1 to 5, appendices C, D + other material
  - 5th Ed: chapters 1 to 5, appendices B, D + other material

  must get; sharing OK

- compare different architectures
  e.g. MIPS and 68000
  Tanenbaum: Structured Computer Organization
  H & P: Computer Architecture: A Quantitative Approach

- hardware emphasis

What is computer architecture?

- architecture

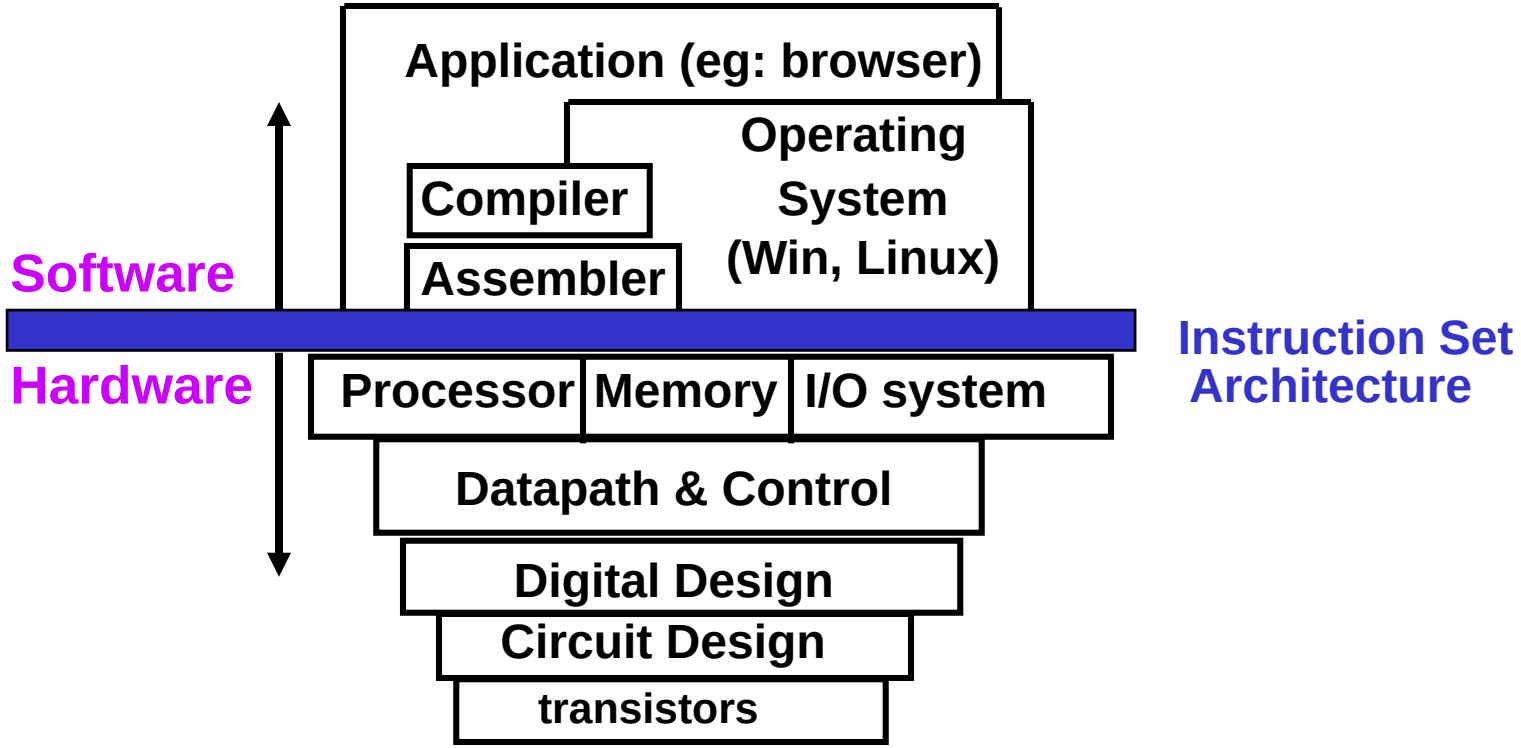
  = instruction set architecture (ISA)

  +

  machine organisation

- ISA examples: P4, ARM, MIPS, SPARC, PowerPC
- instruction set: how abstract? how complex? support for general / special-purpose computing? Compatibility?
- how to choose implementation for a given instruction set?

ISA: between application software and hardware



key: *coordination between levels of abstraction*

Design approaches

- Complex Instruction Set Computers, CISC
  - dense code, simple compiler
  - powerful instruction set, variable format
- Reduced Instruction Set Computers, RISC
  - simple instructions, fixed format, optimising compiler
  - speed, low development cost, adapt to new technology
- ISA level diagram for these?

Instructions: Overview

(3rd Ed: p.48-105, 4th Ed: p.76-139, 5th Ed: p.62-120)

- instruction = opcode what it does
  +
  operand register / memory / data
- MIPS instructions: 3 main types: R, I, J
- design principles for RISCs
  good performance + easy to implement
- use MIPS processor to illustrate ideas in the course

Where are MIPS processors?

Motorola Set-Top Box
DCT6200

Sofaware Security Appliance
Safe@Office 400W

Pioneer DVD Recorder

3com 3102 IP Phone
TI TNETV1050

Maxtor Shared Storage
Device BCM4780

HP Color LaserJet Printer
4100mfp

MIPS architecture

- representative of modern RISC architectures
- 32 registers $0..$31 32 bits each
- $0 wired to 0, the others general-purpose
- register-register or load-store architecture
  - most instructions involves registers only: fast
    `add $1, $2, $3` # reg1 = reg2 + reg3 *comment*
  - special memory access instructions: possibly multicycle
    `lw $8, Astart($19)` # reg8 = M[Astart + reg19]
- goal: minimise memory access; why?

MIPS instructions: R-type

- 3 types: R-type (register)
  I-type (immediate)
  J-type (jump)
  } fixed size: 32 bits

- R-type: arithmetic, comparison, logical, …

```
add $8, $17, $18          # reg8 = reg17 + reg18
```

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0	17	18	8	0	32
opcode	source 1	source 2	dest.	shift	function

- MIPS format: usually destination comes first

MIPS instructions: I-type

- immediate (I-type): memory access
  conditional branches
  arithmetic involving constants

- memory access:

```
lw $8, Astart($19)
```
reg8 = M[Astart + reg19]

6 bits	5 bits	5 bits	16 bits
35	19	8	Astart
opcode	source	dest.	immediate constant

- arithmetic:

```
addi $1, $2, 100
```
reg1 = reg2 + 100

MIPS instructions: J-type

- jump (J-type): unconditional jump to instruction in memory

```
j 1236          # jump to instruction at address 1236
```

6 bits	26 bits
2	1236
opcode	memory[0],[4], … [4,294,967,292]

byte wide

- jal: jump and link # save address of next instruction
  # in register before jumping

- "jump" instructions can be I-type or R-type

I-type : `bne $19,$20,Label` # if reg19 ≠ reg20 goto Label

R-type: `jr  $ra` # jump to address in register ra

Example

- if (i = j) f = g+h; else f = g-h;
- allocate reg16 = f reg17 = g reg18 = h
  reg19 = i reg20 = j
-

```
        bne $19, $20, Else      # if i ≠ j goto Else
        add $16, $17, $18       # f = g+h        (if i = j)
        j   Exit                # goto Exit
Else:   sub $16, $17, $18       # f = g-h        (if i ≠ j)
Exit:
```

- while-loop: similar

Remarks

- only 2 conditional branches, bne and beq
- need slt (set on less than)

```
slt $1, $16, $17    # if reg16 < reg17 then reg1 = 1
                    #                  else  reg1 = 0
```

- implement branch to L on reg16 < reg17 as

```
slt $1, $16, $17    # … if reg1 ≠ 0 then goto L
bne $1, $0, L       #     (reg0 always 0)
```

- load constant hex 000A000B to register 5, use load upper/lower immediate (lui/lli)

```
lui  $5, 10         # reg5 = 000A0000
addi $5, $5, 11     # reg5 = reg5 + 000B
```