Lecture 6 – Processor Architecture Review

Karl R. Wilcox
Karl@cs.rhul.ac.uk

Administration

- **Lab Session to be arranged**
  - **More information next Monday**
- **Lecture Notes**
  - **Are available at http://www.cs.rhul.ac.uk/~karl**
  - **Formats**
    - **Original Powerpoint 2000 slides**
    - **PDF files, A4 2 slides per page, colour**
    - **Let me know if you would like other formats**

Objectives

- **In this lecture we will discuss**
  - **Overview of this part of the course**
  - **Review of Processor Architecture**
    - **From Computer Engineering I**

Course Section Overview

- **In this section of the course we will cover processor design and systems architecture**
  - **Design of pipelined datapaths**
  - **caches**
  - **virtual memory systems**
  - **superscalar processors**
  - **Parallel Systems**

The Course book

- **"Computer Organisation and Design – The Hardware / Software Interface" 2nd Edition**
  - **David A. Patterson and John L. Hennessy**
  - **Morgan Kaufman**
  - **ISBN 1-55860-491-X**

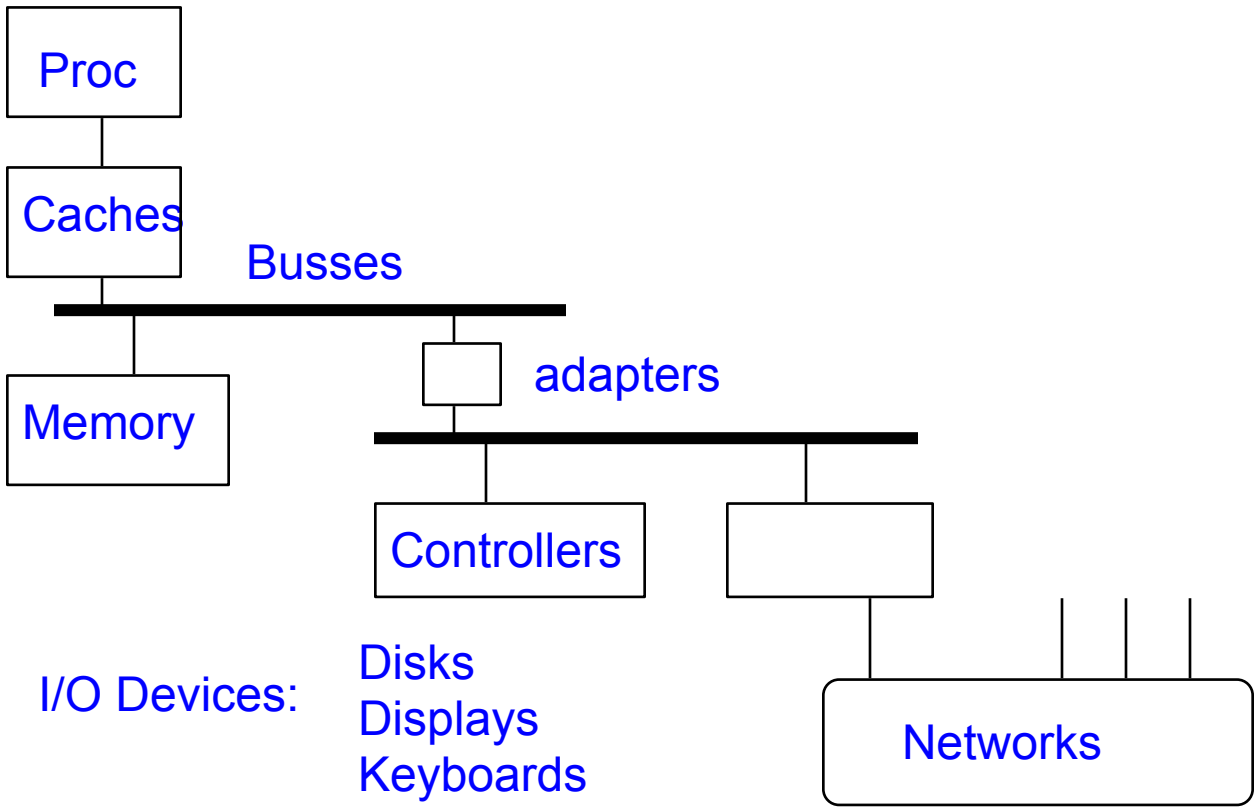
- **Available from the library**
- **Also see http://www.mkp.com/cod2e.htm**

- **Not only covers all of the course material, but is an excellent and interesting book in its own right**

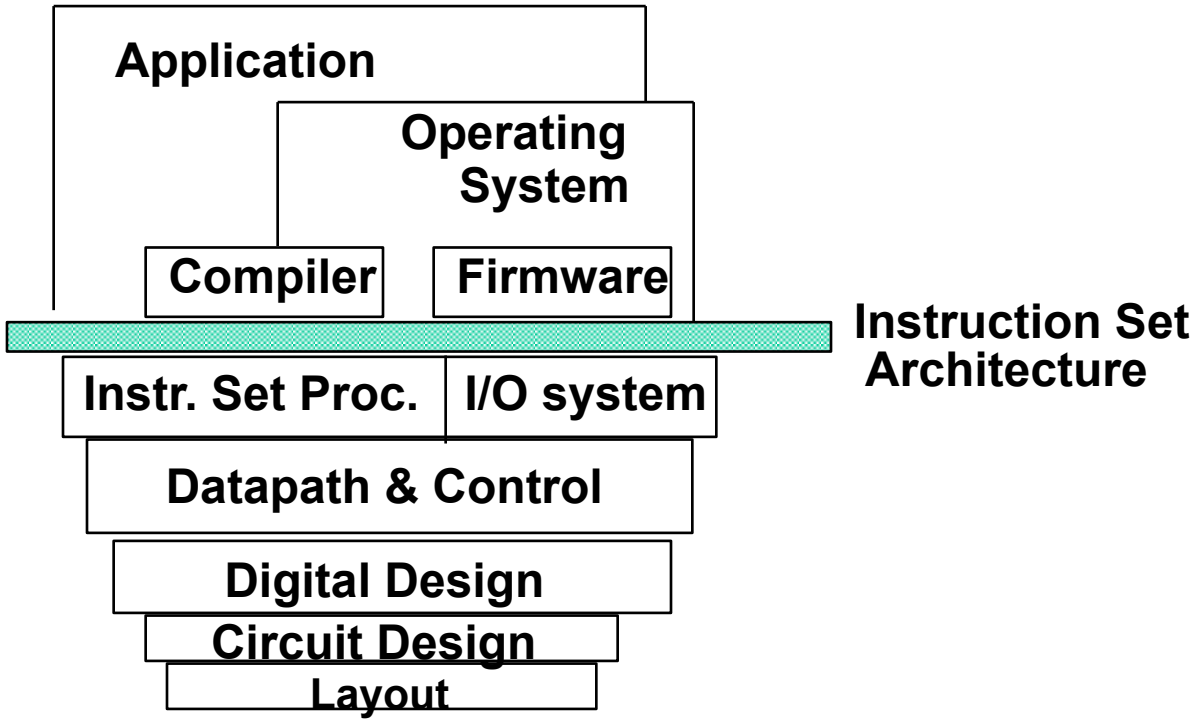
Computer Organisation – Review 1

- **All computers consist of five components**
  - **Processor: (1) datapath and (2) control**
  - **(3) Memory**
  - **(4) Input devices and (5) Output devices**
- **Not all "memory" is created equally**
  - **Cache: fast (expensive) memory are placed closer to the processor**
  - **Main memory: less expensive memory--we can have more**
- **Input and output (I/O) devices have the messiest organization**
  - **Wide range of speed: graphics vs. keyboard**
  - **Wide range of requirements: speed, standard, cost ...**

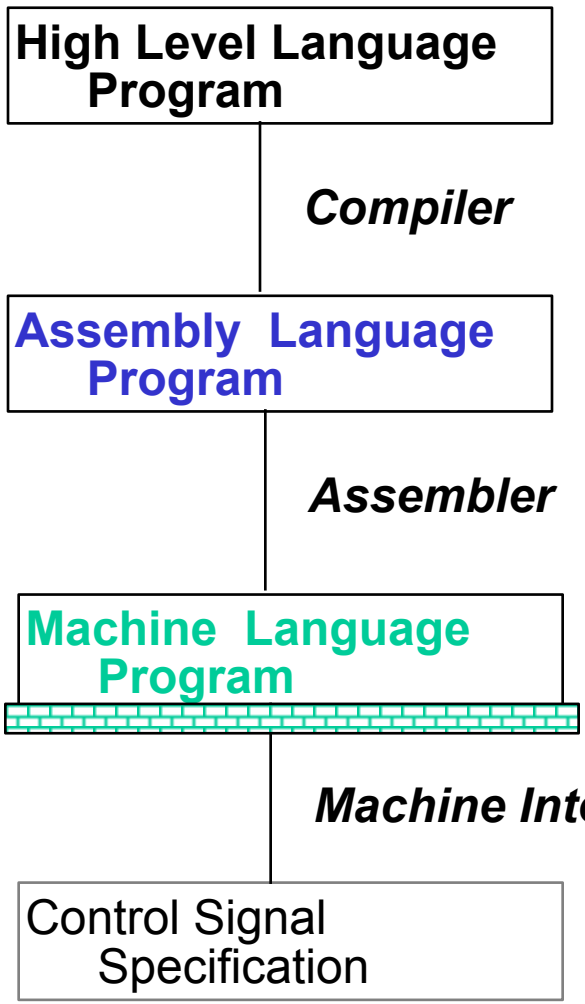
Computer Organisation – Review 2



Levels of Abstraction



Levels of Representation

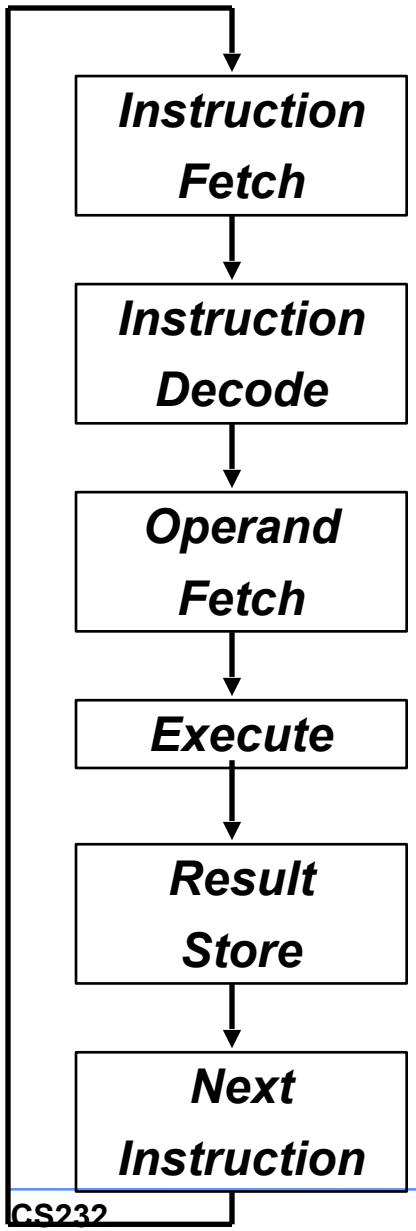



```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

```
lw$15, 0($2)
lw$16, 4($2)
sw     $16, 0($2)
sw     $15, 4($2)
```

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

ALUOP[0:3] <= InstReg[9:11] & MASK



The Instruction Set Architecture

- Instruction Format or Encoding
  - how is it decoded?
- Location of operands and result
  - where other than memory?
  - how many explicit operands?
  - how are memory operands located?
  - which can or cannot be in memory?
- Data type and Size
- Operations
  - what are supported
- Successor instruction
  - jumps, conditions, branches

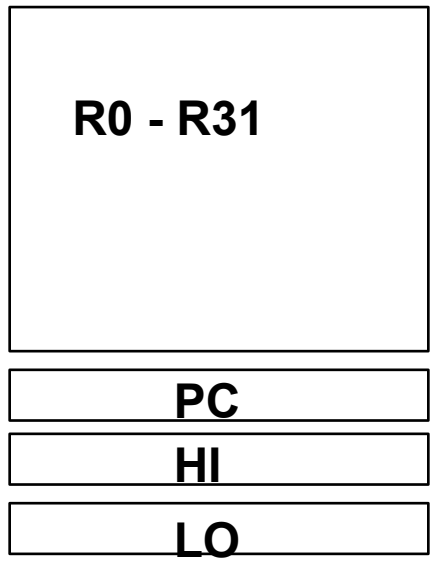
Summary of the MIPS Processor

- **32-bit fixed format inst** (3 formats)
- **32 32-bit GPR** (R0 contains zero) and 32 FP registers (and HI LO)
  - partitioned by software convention
- **3-address, reg-reg arithmetic instr.**
- **Single address mode for load/store:** base+displacement
  - no indirection, scaled
- **16-bit immediate plus LUI**
- **Simple branch conditions**
  - compare against zero or two registers for =,°
  - no integer condition codes
- **Delayed branch**
  - execute instruction after the branch (or jump) even if the branch is taken (Compiler can fill a delayed branch with useful work about 50% of the time)

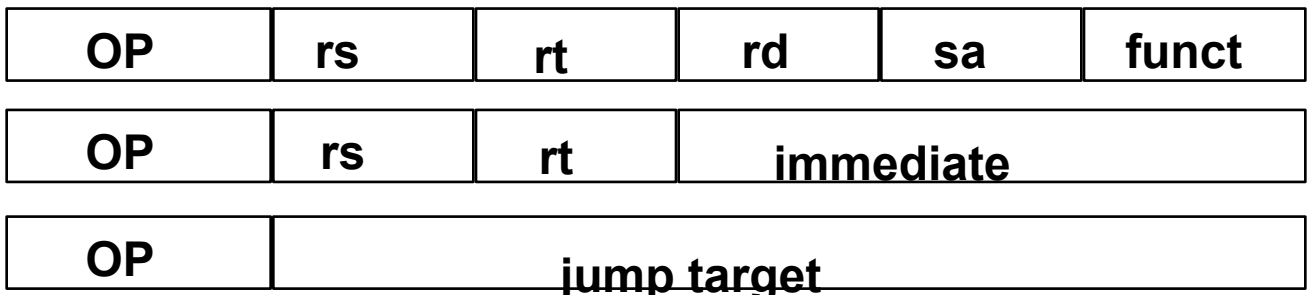
MIPS R3000 Instruction Set Architecture

- **Instruction Categories**
  - Load/Store
  - Computational
  - Jump and Branch
  - Floating Point
    - coprocessor
  - Memory Management
  - Special

Registers



3 Instruction Formats: all 32 bits wide



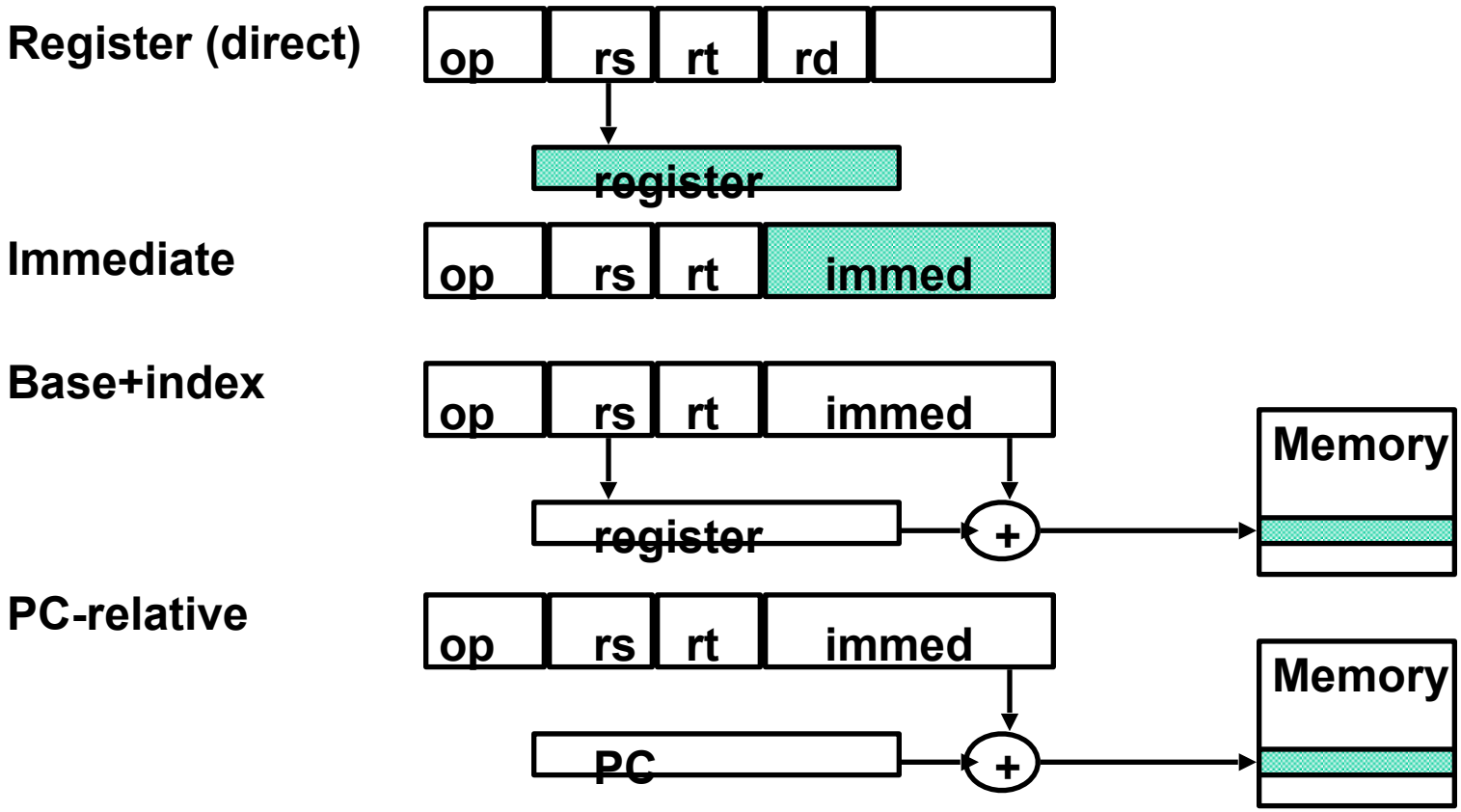
Royal Holloway University of London

MIPS Register Conventions

0	zero	constant 0
1	at	reserved for assembler
2	v0	expression evaluation &
3	v1	function results
4	a0	arguments
5	a1	
6	a2	
7	a3	
8	t0	temporary: caller saves
. . .		(callee can clobber)
15	t7	
16	s0	callee saves
. . .		(caller can clobber)
23	s7	
24	t8	temporary (cont'd)
25	t9	
26	k0	reserved for OS kernel
27	k1	
28	gp	Pointer to global area
29	sp	Stack pointer
30	fp	frame pointer
31	ra	Return Address (HW)

MIPS Addressing Modes

- **All instructions 32 bits wide**



Royal Holloway University of London

MIPS Arithmetic Operations

Instruction	*Example*	*Meaning*	*Comments*
add	add $1,$2,$3	$1 = $2 + $3	3 operands; exception possible
subtract	sub $1,$2,$3	$1 = $2 – $3	3 operands; exception possible
add immediate	addi $1,$2,100	$1 = $2 + 100	+ constant; exception possible
add unsigned	addu $1,$2,$3	$1 = $2 + $3	3 operands; no exceptions
subtract unsigned	subu $1,$2,$3	$1 = $2 – $3	3 operands; no exceptions
add imm. unsign.	addiu $1,$2,100	$1 = $2 + 100	+ constant; no exceptions
multiply	mult $2,$3	Hi, Lo = $2 x $3	64-bit signed product
multiply unsgned	multu$2,$3	Hi, Lo = $2 x $3	64-bit unsigned product
divide	div $2,$3	Lo = $2 ÷ $3, Hi = $2 mod $3	
divide unsigned	divu $2,$3	Lo = $2 ÷ $3, Hi = $2 mod $3	
Move from Hi	mfhi $1	$1 = Hi	Used to get copy of Hi
Move from Lo	mflo $1	$1 = Lo	Used to get copy of Lo

Royal Holloway University of London

MIPS Logical Instructions

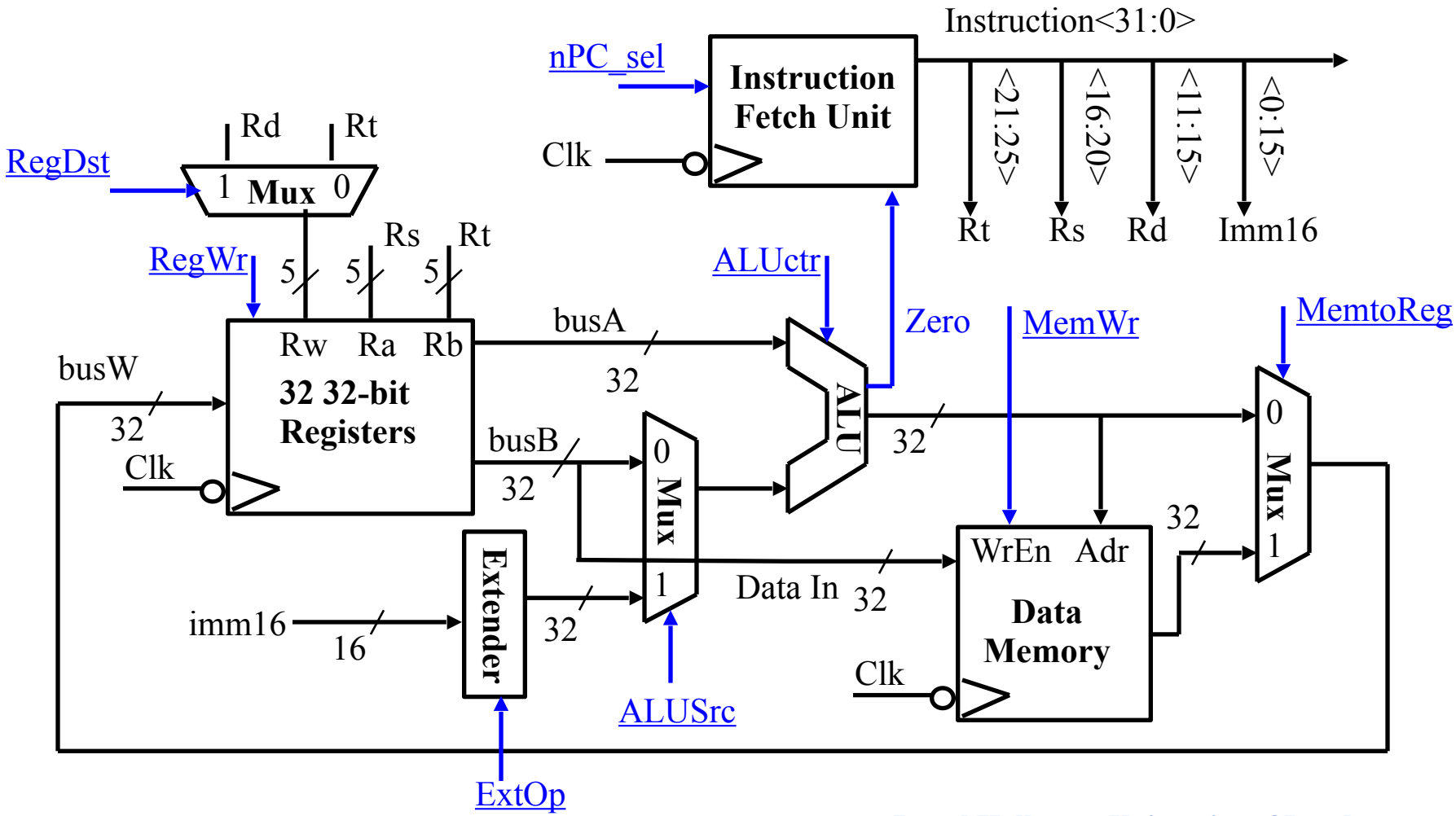
Instruction	*Example*	*Meaning*	*Comment*
and	and $1,$2,$3	$1 = $2 & $3	3 reg. operands; Logical AND
or	or $1,$2,$3	$1 = $2 | $3	3 reg. operands; Logical OR
xor	xor $1,$2,$3	$1 = $2 ⊕ $3	3 reg. operands; Logical XOR
nor	nor $1,$2,$3	$1 = ~($2 |$3)	3 reg. operands; Logical NOR
and immediate	andi $1,$2,10	$1 = $2 & 10	Logical AND reg, constant
or immediate	ori $1,$2,10	$1 = $2 | 10	Logical OR reg, constant
xor immediate	xori $1, $2,10	$1 = ~$2 &~10	Logical XOR reg, constant
shift left logical	sll $1,$2,10	$1 = $2 << 10	Shift left by constant
shift right logical	srl $1,$2,10	$1 = $2 >> 10	Shift right by constant
shift right arithm.	sra $1,$2,10	$1 = $2 >> 10	Shift right (sign extend)
shift left logical	sllv $1,$2,$3	$1 = $2 << $3	Shift left by variable
shift right logical	srlv $1,$2, $3	$1 = $2 >> $3	Shift right by variable
shift right arithm.	srav $1,$2, $3	$1 = $2 >> $3	Shift right arith. by variable

Royal Holloway University of London

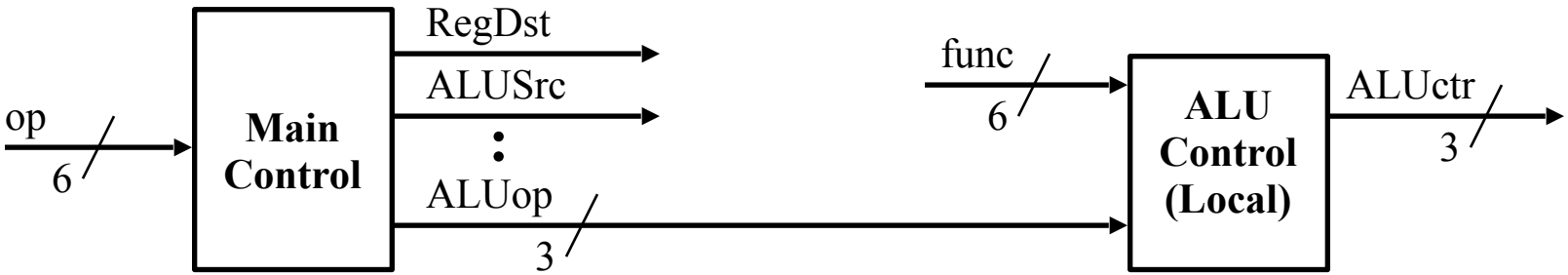
MIPS Comparison Instructions

Instruction	*Example*	*Meaning*
branch on equal	beq $1,$2,100	if ($1 == $2) go to PC+4+100
	Equal test; PC relative branch	
branch on not eq.	bne $1,$2,100	if ($1!= $2) go to PC+4+100
	Not equal test; PC relative	
set on less than	slt $1,$2,$3	if ($2 < $3) $1=1; else $1=0
Compare less than; 2's comp.		
set less than imm.	slti $1,$2,100	if ($2 < 100) $1=1; else $1=0
Compare < constant; 2's comp.		
set less than uns.	sltu $1,$2,$3	if ($2 < $3) $1=1; else $1=0
Compare less than; natural numbers		
set l. t. imm. uns.	sltiu $1,$2,100	if ($2 < 100) $1=1; else $1=0
Compare < constant; natural numbers		
jump	j 10000	go to 10000
	Jump to target address	
jump register	jr $31	go to $31
	For switch, procedure return	
jump and link	jal 10000	$31 = PC + 4; go to 10000
	For procedure call	

Inside the MIPS Processor



Datapath Control Signals

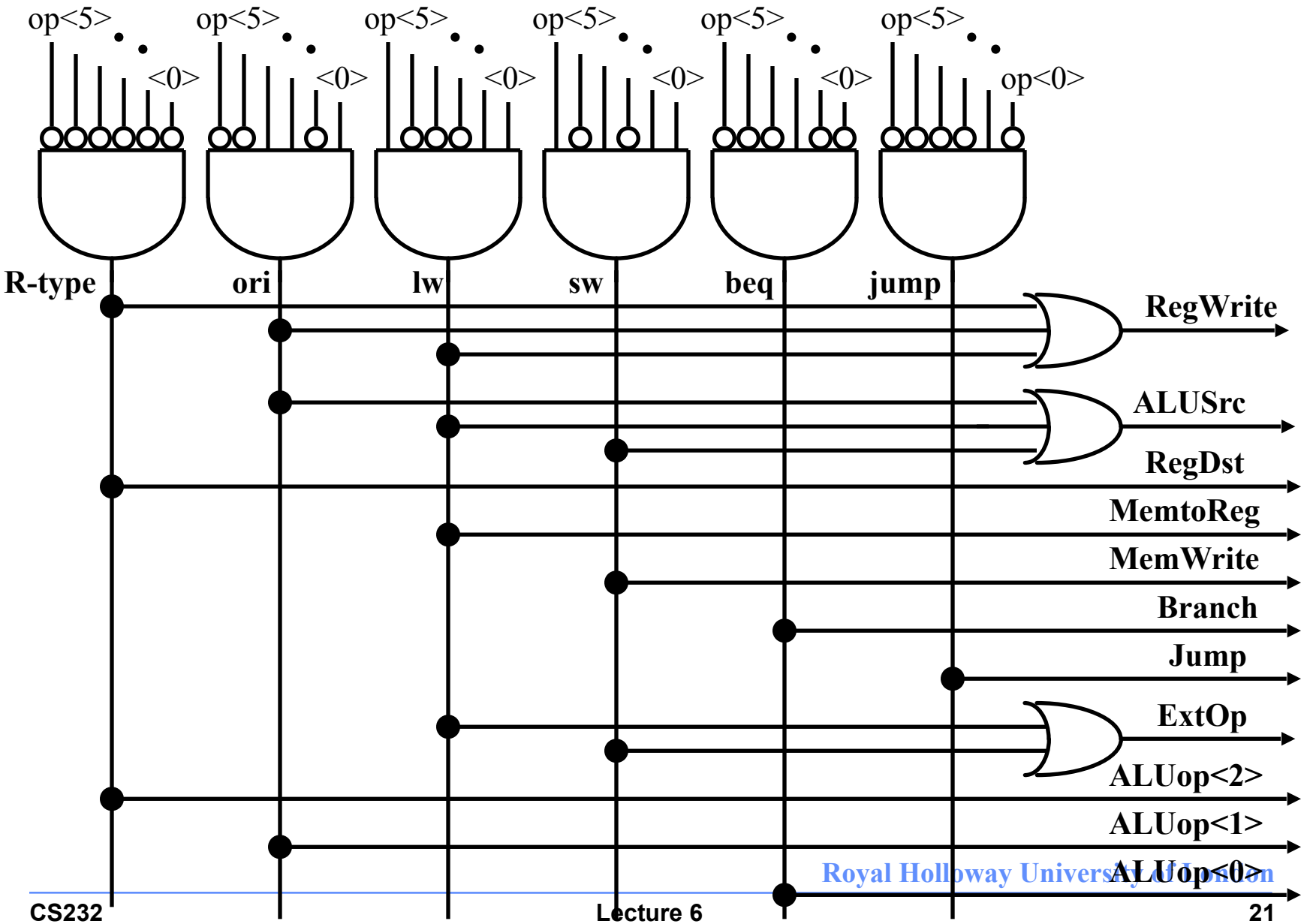


- **On the basis of the opcode (from the instruction) we need to generate appropriate control signals for each of the devices**

Royal Holloway University of London

Truth Table for Main Control

op	00 0000	00 1101	10 0011	10 1011	00 0100	00 0010
	R-type	**ori**	**lw**	**sw**	**beq**	**jump**
RegDst	1	0	0	x	x	x
ALUSrc	0	1	1	1	0	x
MemtoReg	0	0	1	x	x	x
RegWrite	1	1	1	0	0	0
MemWrite	0	0	0	1	0	0
Branch	0	0	0	0	1	0
Jump	0	0	0	0	0	1
ExtOp	x	0	1	1	x	x
ALUop (Symbolic)	"R-type"	Or	Add	Add	Subtract	xxx
ALUop <2>	1	0	0	0	0	x
ALUop <1>	0	1	0	0	0	x
ALUop <0>	0	0	0	0	1	x



Royal Holloway University of London

Summary

- **Review Computer Architecture I !**
- **For this part of the course we are particularly interested in:**
  - **Instruction Set Architecture**
  - **Datapaths**
- **Recall – MIPS architecture and instruction set**
- **Datapath control signals generated by FSM, based on instruction op code**

Next Week

- Further review of datapath
- Introduction to pipelining

- Next Lecture, Monday C103 CS