Basic Processor Design – RISC

- All instructions are the same length
  - typically 1 word (i.e. 16 bits)
- Load/Store Architecture
  - All arithmetic and logic instructions deal only with registers and immediate values[1]

    e.g. `ADD R3,R2,R5` R5 ← R3 + R2

    `OR R3,13,R5` R5 ← R3 | 13
  - Separate instructions are needed for access to locations in memory.

    e.g. `LD [R4+13],R7` R7 ← *mem*(R4 + 13)

    `ST R7,[R4+13]` *mem*(R4 + 13) ← R7

    mem(nnn) is shorthand for the data location in memory with address nnn.
  - Instruction set is maximally orthogonal

[1] an immediate (or literal) value is a data value encoded in the instruction word.

Basic Processor Design – RISC

Q1$_{RISC}$
How many Register Addresses in an Arithmetic/Logic Instruction?

- Usually 2 or 3 for RISC

  2: `ADD Rx,Ry` Rx ← Rx + Ry

  3: `ADD Rx,Ry,Rz` Rz ← Rx + Ry

Q2$_{RISC}$
How many General Purpose Registers?

- Usually $2^n - 1$ for RISC

  This gives 2^n addressable registers including the dummy register, R0[2].

  We then need n bits per register address in the instruction.

  With a 16 bit instruction length, $n = 2$ (i.e. 3 registers + R0) or $n = 3$ (i.e. 7 registers + R0) are sensible values.

[2] R0 is always zero

Basic Processor Design – RISC

Q3$_{RISC}$
How many Bits do we use for Short Immediates?

- Used in instructions like

  `ADD R3,5,R2` R2 ← R3 + 5

  `ST R7,[R4+13]` *mem*(R4 + 13) ← R7

  Sensible values for a 16 bit instruction length are in the range $4 \le s \le 9$ giving 2's complement values in the range $-2^{s-1} \le imm \le 2^{s-1} - 1$

Q3A$_{RISC}$
Do we support All Arithmetic/Logic instructions and All Load/Store instructions in both Register-Register and Register-Immediate forms?

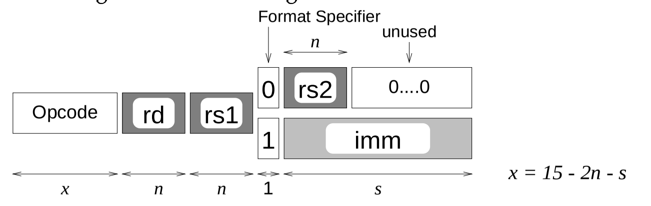
- Some RISC processors support only Register-Immediate form for Load/Store instructions.
- Some RISC processors support Register-Register form for all Arithmetic/Logic instructions and Register-Immediate form for a subset of these instructions.

Basic Processor Design – RISC

Q4$_{RISC}$
What Instruction Fields do we provide? How are they arranged?

- RISC instruction coding is highly orthogonal - any instruction may use any registers.
- Requirement for maximum length short immediate makes RISC coding tight.

Assume Q1$_{RISC}$: 3, Q2$_{RISC}$: $2^n - 1$, Q3$_{RISC}$: s, Q3A$_{RISC}$: YES
A suitable coding for Arithmetic/Logic and Load/Store instructions is:



Example: If $n = 2$ and $s = 7$ then $x = 4$ giving up to 2^x (=16) instructions.

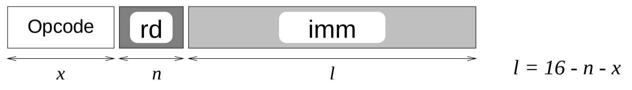
Basic Processor Design – RISC

Most RISC processors support an instruction to set the upper bits of a register. The MIPS processor calls it LUI (load upper immediate) while the SPARC processor calls it SETHI.
For SPARC, the sequence of instructions required to set upper and lower parts of a register is:

`SETHI 200,Rx`	Rx $\leftarrow$ 200 $\times 2^{10}$
`ADD Rx,5,Rx`	Rx $\leftarrow$ Rx + 5

Note that the $\times 2^{10}$ (i.e. shift left by 10) value comes from the SPARC word length (32) less the length of the long immediate used for SETHI (22). In our example it will be $\times 2^{16-l}$ where l is the length of our long immediate.

To code a SETHI or LUI instruction we need fewer fields:

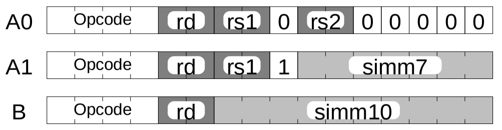


Example: If $n = 2$ and $x = 4$ then $l = 10$.
Note: $s + l \geq 16$ for the SETHI/ADD sequence to produce a 16 bit result.

Basic Processor Design – RISC

Example Coding #1

Assume Q1$_{RISC}$: 3, Q2$_{RISC}$: 3 ($n = 2$), Q3$_{RISC}$: 7 ($s = 7$), Q3A$_{RISC}$: YES
This gives $x = 4$ and $l = 10$ with the coding shown below[3]:



In this example coding up to 16 instructions are supported, each of which supports either coding A (i.e. A0 and A1) or coding B.

This coding is loosely based on that of the SPARC processor family.

[3]note that in this example short and long immediates (`simm7` and `simm10`) are signed numbers

Basic Processor Design – RISC

Q5$_{RISC}$ What Instructions will we support?

Type	Function	Mnemonic	Function	Mnemonic
Arithmetic	Add	ADD	Subtract	SUB
	Add with Carry	ADDX	Subtract with Borrow	SUBX
Logic	Bitwise AND	AND	Bitwise OR	OR
	Bitwise Exclusive OR	XOR	Logical Shift Right	LSR
Data Movement	Load	LD	Store	ST
	Set High	SETHI		
Control Transfer	Branch if Zero	BZ	Branch to Subroutine	BSR
	Jump and Link	JMPL		

This set has been chosen based on a maximum of 16 instructions (to match example coding #1), with support for a complete set of common arithmetic and logical functions (note that multiply is too complex to be included, while shift left is accomplished by adding a number to itself).
The control transfer functions are a minimum set to support subroutines. BZ provides a conditional PC relative branch (unconditional if R0 is tested). BSR provides a PC relative branch which stores the calling address in a register. JMPL provides the ability to return from a subroutine and also the ability to jump a long way.

Basic Processor Design – RISC

Q5A$_{RISC}$ Define Assembly Language Syntax[4] and Semantics.

	Mnemonic	Format	Syntax	Semantics
Arithmetic	ADD*	A	`ADD Rs1,Op2,Rd`	$Rd \leftarrow Rs1 + Op2$
			where `Op2` is either `Rs2` or `simm7`	
	SUB*	A	`SUB Rs1,Op2,Rd`	$Rd \leftarrow Rs1 - Op2$
	ADDX*	A	`ADDX Rs1,Op2,Rd`	$Rd \leftarrow Rs1 + Op2 + C$
	SUBX*	A	`SUBX Rs1,Op2,Rd`	$Rd \leftarrow Rs1 - Op2 - C$
Logic	AND	A	`AND Rs1,Op2,Rd`	$Rd \leftarrow Rs1 \,\&\, Op2$
	OR	A	`OR Rs1,Op2,Rd`	$Rd \leftarrow Rs1 \mid Op2$
	XOR	A	`XOR Rs1,Op2,Rd`	$Rd \leftarrow Rs1$ ^ $Op2$
	LSR*	A	`LSR Rs1,Rd`	$Rd \leftarrow Rs1 >> 1$
Data Movement	LD	A	`LD [Rs1+Op2],Rd`	$Rd \leftarrow mem(Rs1 + Op2)$
	ST	A	`ST Rd,[Rs1+Op2]`	$mem(Rs1 + Op2) \leftarrow Rd$
	SETHI	B	`SETHI simm10,Rd`	$Rd \leftarrow simm10 << 6$
Control Transfer	BZ	B	`BZ Rd,simm10`	$if\ (Rd = 0)\ then\ PC \leftarrow PC + simm10$
	BSR	B	`BSR simm10,Rd`	$Rd \leftarrow PC;\ PC \leftarrow PC + simm10$
	JMPL	A	`JMPL Rs1+Op2,Rd`	$Rd \leftarrow PC;\ PC \leftarrow Rs1 + Op2$

[4]operand order follows SPARC convention

*marked commands update the Carry flag (C)

Basic Processor Design – RISC

$Q5B_{RISC}$
What Opcodes will be Assigned?

Op[3:2] \ Op[1:0]	00	01	11	10	
00	ADD	ADDX	AND	OR	Format A
01	SUB	SUBX	XOR	LSR	Format A
11	LD	ST	-	JMPL	Format A
10	SETHI	-	BZ	BSR	Format B

Instructions are grouped so that decoding is simple.