

# Computer Instructions

CSE 410 - Computer Systems  
October 3, 2001

## Readings and References

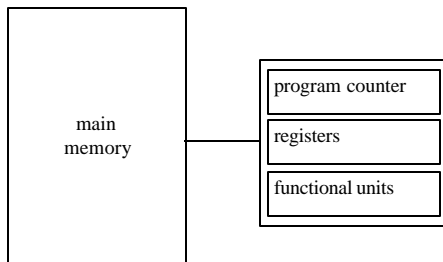
- Reading
  - Sections 3.1-3.4, Patterson and Hennessy, Computer Organization & Design
- Other References
  - D Sweetman, *See MIPS Run*, Morgan Kaufman, Publishers
    - section 8.5, Instruction encoding
    - section 11.6, Endianess

3-Oct-2001

CSE 410 - Computer Instructions

2

## A very simple organization



3-Oct-2001

CSE 410 - Computer Instructions

3

## Instructions in main memory

- Instructions are stored in main memory
- Program counter (PC) points to the next instruction
  - All MIPS instructions are 4 bytes long, and so instruction addresses are always multiples of 4
- Program addresses are 32 bits
  - $2^{32} = 4,294,967,296 = 4 \text{ GigaBytes (GB)}$

3-Oct-2001

CSE 410 - Computer Instructions

4

## Instructions in memory

	:	:	:	:	:	:
20						
16						
12						
8						
4					instruction value	
0					instruction value	

3-Oct-2001

CSE 410 - Computer Instructions

5

## Some common storage units

<i>unit</i>	<i># bits</i>					
byte	8	<table><tr><td></td></tr></table>				
half-word	16	<table><tr><td></td><td></td></tr></table>				
word	32	<table><tr><td></td><td></td><td></td><td></td></tr></table>				
double word	64	<table><tr><td></td><td></td><td></td><td></td></tr></table>				

3-Oct-2001

CSE 410 - Computer Instructions

6

## Alignment

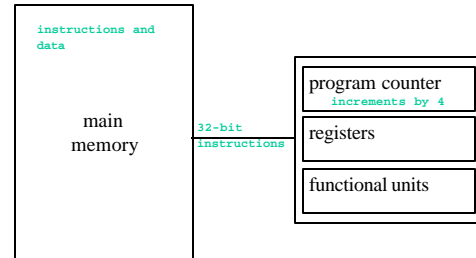
- An object in memory is “aligned” when its address is a multiple of its size
- Byte: always aligned
- Halfword: address is multiple of 2
- Word: address is multiple of 4
- Double word: address is multiple of 8
- Alignment simplifies load/store hardware

3-Oct-2001

CSE 410 - Computer Instructions

7

## System organization so far



3-Oct-2001

CSE 410 - Computer Instructions

8

## Registers

- 32 bits wide
  - 32 bits is 4 bytes
  - same as a word in memory
  - signed values from  $-2^{31}$  to  $+2^{31}-1$
  - unsigned values from 0 to  $2^{32}-1$
- easy to access and manipulate
  - on chip, so very fast to access
  - 32 registers, so easy to address

3-Oct-2001

CSE 410 - Computer Instructions

9

## Register addresses

- 32 general purpose registers
- how many bits does it take to identify a register?
  - 5 bits, because  $2^5 = 32$
- 32 registers is a compromise selection
  - more would require more bits to identify
  - fewer would be harder to use efficiently

3-Oct-2001

CSE 410 - Computer Instructions

10

## Register numbers and names

number	name	usage
0	<b>zero</b>	always returns 0
1	<b>at</b>	reserved for use as assembler temporary
2-3	<b>v0, v1</b>	values returned by procedures
4-7	<b>a0-a3</b>	first few procedure arguments
8-15, 24, 25	<b>t0-t9</b>	temps - can use without saving
16-23	<b>s0-s7</b>	temps - must save before using
26, 27	<b>k0, k1</b>	reserved for kernel use - may change at any time
28	<b>gp</b>	global pointer
29	<b>sp</b>	stack pointer
30	<b>fp</b> or <b>s8</b>	frame pointer
31	<b>ra</b>	return address from procedure

3-Oct-2001

CSE 410 - Computer Instructions

11

## How are registers used?

- Many instructions use 3 registers
  - 2 source registers
  - 1 destination register
- For example
  - **add \$t1, \$a0, \$t0**
    - add a0 and t0 and put result in t1
  - **add \$t1, \$zero, \$a0**
    - move contents of a0 to t1 ( $t1 = 0 + a0$ )

3-Oct-2001

CSE 410 - Computer Instructions

12

## R-format instructions: 3 registers

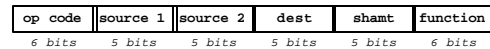
- 32 bits available in the instruction
- 15 bits for the 5-bit register numbers
- The remaining 17 bits are available for specifying the instruction
  - 6-bit op code - basic instruction identifier
  - 5-bit shift amount
  - 6-bit function code

3-Oct-2001

CSE 410 - Computer Instructions

13

## R-format fields



- some common R-format instructions
  - arithmetic: **add, sub, mult, div**
  - logical: **and, or, sll, srl**
  - comparison: **slt** (set on less than)
  - jump through register: **jr**

3-Oct-2001

CSE 410 - Computer Instructions

14

## Bits are just bits

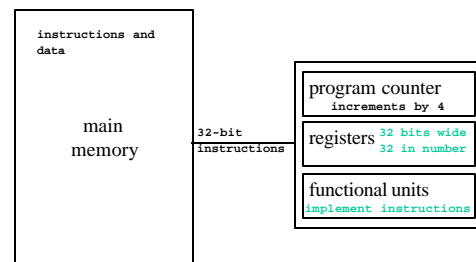
- The bits mean whatever the designer says they mean when the ISA is defined
- How many possible 3-register instructions are there?
  - $2^{17} = 131,072$
  - includes all values of op code, shamt, function
- As the ISA develops over the years, the encoding tends to become less logical

3-Oct-2001

CSE 410 - Computer Instructions

15

## System organization again



3-Oct-2001

CSE 410 - Computer Instructions

16

## Transfer from memory to register

- Load instructions
  - word: `lw rt, address`
  - half word: `lh rt, address`  
`lhu rt, address`
  - byte: `lb rt, address`  
`lbu rt, address`
- signed load => sign bit is extended into the upper bits of destination register
- unsigned load => 0 in upper bits of register

3-Oct-2001

CSE 410 - Computer Instructions

17

## Transfer from register to memory

- Store instructions
  - word: `sw rt, address`
  - half word: `sh rt, address`
  - byte: `sb rt, address`

3-Oct-2001

CSE 410 - Computer Instructions

18

## The “address” term

- There is one basic addressing mode:  
offset + base register value
- Offset is 16 bits ( $\pm 32$  KB)
- Load word pointed to by s0, add t1, store
 

```
lw    $t0, 0($s0)
add   $t0, $t0, $t1
sw    $t0, 0($s0)
```

3-Oct-2001

CSE 410 - Computer Instructions

19

## I-format fields



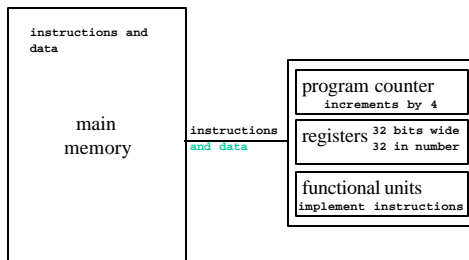
- The contents of the base register and the offset value are added together to generate the address for the memory reference
- Can also use the 16 bits to specify an immediate value, rather than an address

3-Oct-2001

CSE 410 - Computer Instructions

20

## Instructions and Data flow



3-Oct-2001

CSE 410 - Computer Instructions

21

## The eye of the beholder

- Bit patterns have no inherent meaning
- A 32-bit word can be seen as
  - a signed integer ( $\pm 2$  Billion)
  - an unsigned integer or address pointer (0 to 4B)
  - a single precision floating point number
  - four 1-byte characters
  - an instruction

3-Oct-2001

CSE 410 - Computer Instructions

22

## Big-endian, little-endian

- A 32-bit word in memory is 4 bytes long
- but which byte is which address?
- Consider the 32-bit number 0x01234567
  - four bytes: 01, 23, 45, 67
  - most significant bits are 0x01
  - least significant bits are 0x67

3-Oct-2001

CSE 410 - Computer Instructions

23

## Data in memory- big endian

Big endian - **most** significant bits are in byte 0 of the word

⋮	⋮	⋮	⋮	⋮	⋮	byte #	contents
12						7	67
8						6	45
4	01	23	45	67		5	23
0						4	01
	0	1	2	3		← byte offsets	

3-Oct-2001

CSE 410 - Computer Instructions

24

## Data in memory- little endian

Little endian - **least** significant bits are in byte 0 of the word

byte #	contents
7	01
6	23
5	45
4	67

...	...	...	...	...
12				
8				
4	01	23	45	67
0				

3      2      1      0   ← byte offsets

3-Oct-2001

CSE 410 - Computer Instructions

25

## Unsigned binary numbers

- Each bit represents a power of 2
- For unsigned numbers in a fixed width field
  - the minimum value is 0
  - the maximum value is  $2^n - 1$ , where n is the number of bits in the field
- Fixed field widths determine many limits
  - 5 bits = 32 possible values ( $2^5 = 32$ )
  - 10 bits = 1024 possible values ( $2^{10} = 1024$ )

3-Oct-2001

CSE 410 - Computer Instructions

26

## Binary, Hex, and Decimal

	$2^8=256$	$2^7=128$	$2^6=64$	$2^5=32$	$2^4=16$	$2^3=8$	$2^2=4$	$2^1=2$	$2^0=1$	Hex <sub>16</sub>	Decimal <sub>10</sub>
						1	0	0	1	0x3	3
						1	0	0	1	0x9	9
						1	0	1	0	0xA	10
						1	1	1	1	0xF	15
				1	0	0	0	0	0	0x10	16
				1	1	1	1	1	1	0x1F	31
		1	1	1	1	1	1	1	1	0x7F	127
	1	1	1	1	1	1	1	1	1	0xFF	255

3-Oct-2001

CSE 410 - Computer Instructions

27