CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Instructors:
Randy H. Katz
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/fa10

Agenda

- Review
- Operands of the Computer
- Memory and Addresses
- Administrivia + The secret to getting good grades at Berkeley
- Technology Break
- Constants
- Logical Operation
- (if time) Communicating with people
- Summary

Review from Last Lecture

- Course is now Great Ideas in Computer Architecture
  - Layers of Representation/Interpretation
  - Moore's Law
  - Principle of Locality/Memory Hierarchy
  - Parallelism
  - Performance Improvement
- Computer commands called instructions
- Computer vocabulary is called instruction set
  - MIPS is example instruction set in this class
  - add, sub, mult, div
- Rigid format: 1 operation, 2 source operands, 1 destination operand
- 1st Design Principle: Simplicity favors regularity

Operands of Computer Hardware

- High-Level Programming languages: could have millions of variables
- Instruction sets have fixed, smaller number
- Called registers
  - The brick of computer hardware
  - Used to construct computer hardware
  - Visible to programmer
- MIPS Instruction Set has 32 registers

Why Just 32 Registers?

- Design Principle 2: Smaller is faster.
- Hardware would likely be slower with 64, 128, or 256 registers
- 32 is enough for compiler to translate typical C and Java programs are not run out of registers very often
  - ARM instruction set has 16 registers; may be faster, but compiler may run out of registers too often

Names of MIPS Registers

- For registers that hold programmer variables: **`$s0`**, **`$s1`**, **`$s2`**, …
- For registers that hold temporary variables: **`$t0`**, **`$t1`**, **`$t2`**, …
- Suppose variables **`f`**, **`g`**, **`h`**, **`i`**, and **`j`** are assigned to the registers **`$s0`**, **`$s1`**, **`$s2`**, **`$s3`**, and **`$s4`**, respectively. What is MIPS for

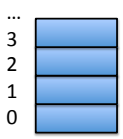
```
f = (g + h) – (i + j);
```

Size of Registers

- A bit is atom of Computer Hardware: contains either 0 or 1
  - As we shall see, the real "alphabet" of computer hardware is 0, 1 so we will eventually express MIPS instructions as combinations of 0s and 1s
- MIPS registers are 32 bits wide
- MIPS calls this quantity a word
  - Some computers use 16-bit wide words, like Intel 80x86

What about Data Structures versus Simple Variables?

- In addition to registers, a computer also has memory that holds millions / billions of words
- Memory is a single dimension array, starting at 0
- To access memory need an address (index)
- But MIPS instructions only operate on registers!
- Solution: instructions that just transfer words (data) between memory and registers
- Called data transfer instructions



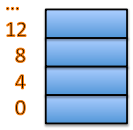
Transfer from Memory to Register?

- MIPS instruction is Load Word, abbreviated `lw`
- Assume that A is an array of 100 words, variables g and h map to registers $s1 and $s2, the starting address, or base address, of the array A is in $s3.
- int A[100];
  g = h + A[8];
- Turns into

```
lw $t0, 8($s3)     # Temporary reg $t0 gets A[8]
add $s1, $s2, $t0  # g = h + A[8]
```


Memory Addresses are in Bytes

- As we shall see, lots of data is smaller than 32 bits but rarely smaller than 8 bits, so works fine if everything is a multiple of 8 bits
- 8 bit item is called a byte (1 word = 4 bytes)
- Memory addresses are really in bytes, not words
- So array indices are in bytes;
  - Word addresses are 4 bytes apart



Transfer from Memory to Register?

- MIPS instruction is Load Word, abbreviated `lw`
- Assume that A is an array of 100 words, variables g and h map to registers $s1 and $s2, the starting address, or base address, of the array A is in $s3.
  g = h + A[8];
- Turns into

```
lw $t0, 32($s3)    # Temporary reg $t0 gets A[8]
add $s1, $s2, $t0  # g = h + A[8]
```


Transfer from Register to Memory?

- MIPS instruction is Store Word, abbreviated `sw`
- Assume that A is an array of 100 words, variables g and h map to registers $s1 and $s2, the starting address, or base address, of the array A is in $s3.
  A[12] = h + A[8];
- Turns into

```
lw $t0, 32($s3)    # Temporary reg $t0 gets A[8]
add $t0, $s2, $t0  # t0 = h + A[8]
                   # A[12] = h + A[8]
```

Speed of Registers vs. Memory?

- Given that
  - Registers: 32 words (128 Bytes)
  - Memory: Billions of bytes (2 GB to 8 GB on laptop)
- and 2nd principle is
  - Smaller is faster
- How much faster are registers than memory??
- About 100-500 times faster!

Course Information

- Instructors:
  - Randy Katz, Dave Patterson
- Teaching Assistants:
  - Andrew Gearhart, Michael Greenbaum, Conor Hughes, Charles Reiss
- Textbooks:
  - Patterson, Hennessey, *Computer Organization and Design*, 4th Edition
  - Kernighan, Ritchie, *The C Programming Language*, 2nd Edition
- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/fa10
- Google Group: 61CFall2010UCB
- Homework 1 Due Friday Midnight

YOUR BRAIN ON COMPUTERS; Hooked on Gadgets, and Paying a Mental Price

NY Times, June 7, 2010, by Matt Richtel

SAN FRANCISCO -- When one of the most important e-mail messages of his life landed in his in-box a few years ago, Kord Campbell overlooked it.

Not just for a day or two, but 12 days. He finally saw it while sifting through old messages: a big company wanted to buy his Internet start-up.

"I stood up from my desk and said, 'Oh my God, oh my God, oh my God,' " Mr. Campbell said. "It's kind of hard to miss an e-mail like that, but I did."

The message had slipped by him amid an electronic flood: two computer screens alive with e-mail, instant messages, online chats, a Web browser and the computer code he was writing.

While he managed to salvage the $1.3 million deal after apologizing to his suitor, Mr. Campbell continues to struggle with the effects of the deluge of data. Even after he unplugs, he craves the stimulation he gets from his electronic gadgets. He forgets things like dinner plans, and he has trouble focusing on his family. His wife, Brenda, complains, "It seems like he can no longer be fully in the moment."

This is your brain on computers.

Scientists say juggling e-mail, phone calls and other incoming information can change how people think and behave. They say our ability to focus is being undermined by bursts of information.

These play to a primitive impulse to respond to immediate opportunities and threats. The stimulation provokes excitement -- a dopamine squirt -- that researchers say can be addictive. In its absence, people feel bored.

The resulting distractions can have deadly consequences, as when cellphone-wielding drivers and train engineers cause wrecks. And for millions of people like Mr. Campbell, these urges can inflict nicks and cuts on creativity and deep thought, interrupting work and family life.

While many people say multitasking makes them more productive, research shows otherwise. Heavy multitaskers actually have more trouble focusing and shutting out irrelevant information, scientists say, and they experience more stress.

And scientists are discovering that even after the multitasking ends, fractured thinking and lack of focus persist. In other words, this is also your brain off computers.

The Rules (and we really mean it!)

The Secret to Getting Good Grades

- Grad student said he figured finally it out
  - (Mike Dahlin, now Professor at UT Texas)
- What is the secret?
- Do assigned reading night before, so that get more value from lecture

Agenda

- Review
- Operands of the Computer
- Memory and Addresses
- Administrivia + The secret to getting good grades at Berkeley
- Technology Break
- Constants
- Logical Operation
- (if time) Communicating with people
- Summary

Constants as Operands

- Many programs use constant numbers
  - E.g., Increment for a loop, index in a data structure
  - >50% of operands are constants in MIPS programs
- So far, would have to be in memory
- To add the constant 4 to register $s3

```
lw $t0, AddrConstant4($s1)  # $t0 = constant 4
add $s3, $s3, $t0           # $s3 = $s3 + $t0 ($t0 == 4)
```

assuming that $s1 + AddrConstant4 is the memory address of the constant 4

Constants as Operands

- Can't we get constants without using a load word instruction, since they are so widely used?
- Design Principle 3: Make the common case fast.
- Idea: offer new versions of arithmetic instructions where one operand is a constant
- Called *add immediate* or *addi*
- To add 4 to register $s3:

```
addi  $s3,$s3,4   # $s3 = $s3 + 4
```

- Avoids 100X slowdown of memory access

How translate Java to MIPS?

- A program called a compiler translates from language programmers write (Java) into language computer understands (MIPS) before the program is run on the hardware
  - Act of translating called compiling
- Brings in concepts of
- Compile-time: what compiler does to program before it is run
- Run-time: what program does during execution

MIPS Logical Instructions

- Useful to operate on fields of bits within a word
  - e.g., characters within a word (8 bits)
- Operations to pack /unpack bits into words
- Called logical operations

Logical operations	C operators	Java operators
Bit-by-bit AND	&	&
Bit-by-bit OR	|	|
Shift left	<<	<<
Shift right	>>	>>>

Guess the other MIPS instructions

- Java operator &: **$s1** & **$s2** and put result in **$s0**?
- Java operator |: **$s1** | **$s2** and put result in **$s0**?
- Java operator <<: **$s1** << **$s2** and put result in **$s0**?
- Java operator >>: **$s1** >> **$s2** and put result in **$s0**?

MIPS Logical Instructions

- Useful to operate on fields of bits within a word
  - e.g., characters within a word (8 bits)
- Operations to pack /unpack bits into words
- Called logical operations

Logical operations	C operators	Java operators	MIPS instructions
Bit-by-bit AND	&	&	**and**
Bit-by-bit OR	|	|	**or**
Shift left	<<	<<	**sll**
Shift right	>>	>>>	**srl**

Constants as Operands

- Immediate version of many instructions
- And immediate *andi*
- Or immediate *ori*
- Shift immediate *sll*, *srl*
  - Assumption that normally want to define a constant shift amount
  - Variable shifts using *sllv*, *srlv*
- Why no subtract immediate?

Zero as a Special Constant

- For reasons we shall see, 0 is such a popular constant that MIPS reserves an register to always have the value 0
- $zero

Communicating with People

- Computers were invented to crunch numbers, but soon needed text to interact with people
- Needed representation of English alphabet
- How many symbols do you need?
- How many bits for that many symbols?

Communicating with People

- American Standard Code for Information Interchange (ASCII)
  - 8 bits or 1 byte per symbol

ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character
32	space	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	DEL

What about Non-English Alphabet Languages?

- Unicode () tried to capture all languages, include long gone ones; below are example alphabets!

Latin	Malayalam	Tagbanwa	General Punctuation
Greek	Sinhala	Khmer	Spacing Modifier Letters
Cyrillic	Thai	Mongolian	Currency Symbols
Armenian	Lao	Limbu	Combining Diacritical Marks
Hebrew	Tibetan	Tai Le	Combining Marks for Symbols
Arabic	Myanmar	Kangxi Radicals	Superscripts and Subscripts
Syriac	Georgian	Hiragana	Number Forms
Thaana	Hangul Jamo	Katakana	Mathematical Operators
Devanagari	Ethiopic	Bopomofo	Mathematical Alphanumeric Symbols
Bengali	Cherokee	Kanbun	Braille Patterns
Gurmukhi	Unified Canadian Aboriginal Syllabic	Shavian	Optical Character Recognition
Gujarati	Ogham	Osmanya	Byzantine Musical Symbols
Oriya	Runic	Cypriot Syllabary	Musical Symbols
Tamil	Tagalog	Tai Xuan Jing Symbols	Arrows
Telugu	Hanunoo	Yijing Hexagram Symbols	Box Drawing
Kannada	Buhid	Aegean Numbers	Geometric Shapes

What about Non-English Alphabet Languages?

- How many bits to represent all human languages?
- 16 bits or 2 bytes per symbol (originally)
- 16 bits called *halfword* in MIPS
- ASCII encoding is subset of UNICODE
  - 1st byte 0, 2nd byte is ASCII

Summary

- Operands are limited in instruction sets: registers
  - 32 registers in MIPS
  - 2nd Design Principle: Smaller is faster
  - Each 32 bits wide ("word")
  - They are 100-500 times faster than word in memory
- Constants also operands
  - 3rd Design Principle : Make the common case fast.
  - Variation of instructions: addi, multi, divi,
- Computers do logical operations as well as arithmetic
  - and, or, shift left (sll), shift right (srl), andi, ori,
- Characters: 8-bit ASCII and 16-bit Unicode
- From Java to MIPS instructions called compiling
  - Compile-time (before program run) vs. run-time