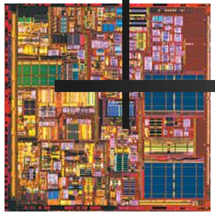
ECE 411
Fall 2015

Lecture 3



ISA Design;
Performance Metrics

Key ISA decisions

Instruction format – Length? Variable? Fixed? Fields?

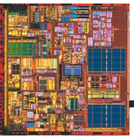
How many registers?

Where do instruction operands reside?

- e.g., can you add contents of memory to a register?

Operands

- how many? how big?
- how are memory addresses computed?



Where do operands reside (*when the ALU needs them*)?

Stack machine:

- "Push" loads memory into 1st register ("top of stack"), moves other regs down
- "Pop" does the reverse.
- "Add" combines contents of first two registers, moves rest up.

Accumulator machine:

- Only 1 register (called the "accumulator")
- Instruction include "store" and "acc ← acc + mem"

Register-Memory machine :

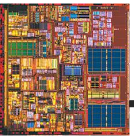
- Arithmetic instructions can use data in registers and/or memory

Load-Store Machine (aka Register-Register Machine):

- Arithmetic instructions can only use data in registers.

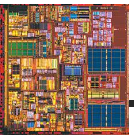
ILLINOIS

Comparing the ISA classes



Code sequence for `C = A + B`

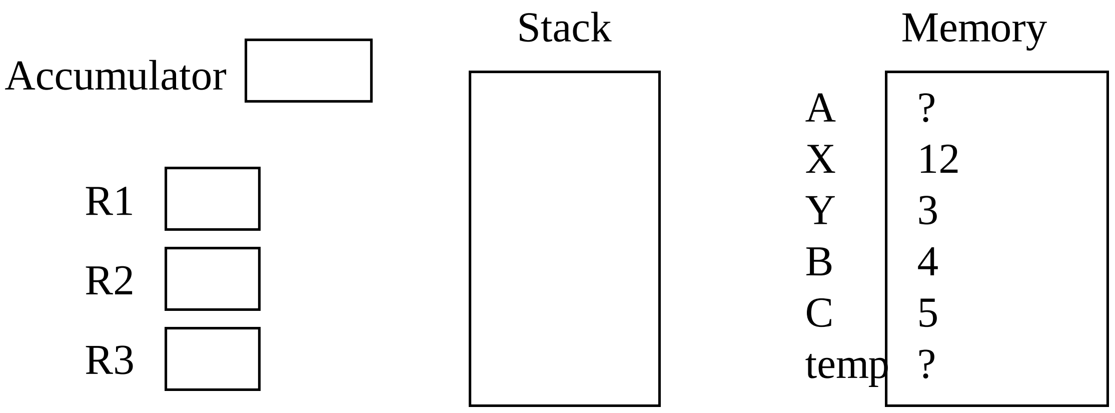
Stack	Accumulator	Register-Memory	Load-Store
`Push A`	`Load A`	`Add C, A, B`	`Load R1,A`
`Push B`	`Add B`		`Load R2,B`
`Add`	`Store C`		`Add R3,R1,R2`
`Pop C`			`Store C,R3`
Java VMs	DSPs	VAX, x86 partially	

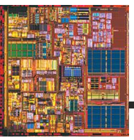


Comparing the ISA classes

Stack Accumulator Reg-Mem Load-store

$$\boxed{A = X*Y + X*Z}$$





Instruction formats *-what does each bit mean?*

Machine needs to determine quickly,

- “This is a 6-byte instruction”
- “Bits 7-11 specify a register”
- ...
- Serial decoding bad

Having many different instruction formats...

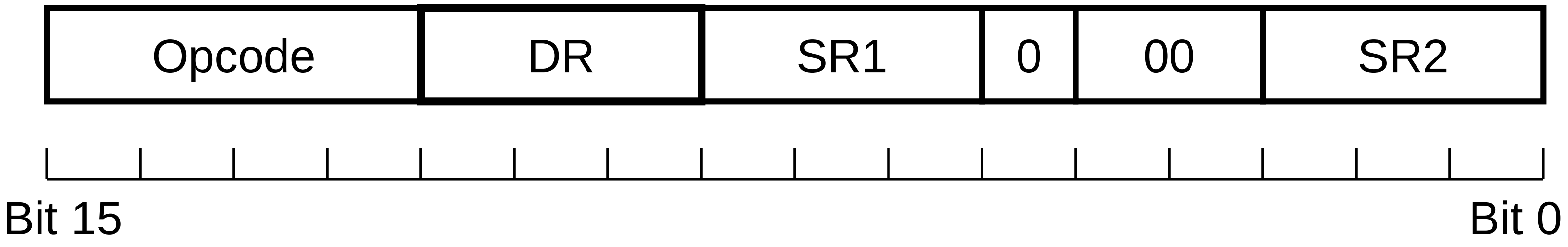
- complicates decoding
- uses instruction bits (to specify the format)

What would be a good thing about having many different instruction formats?

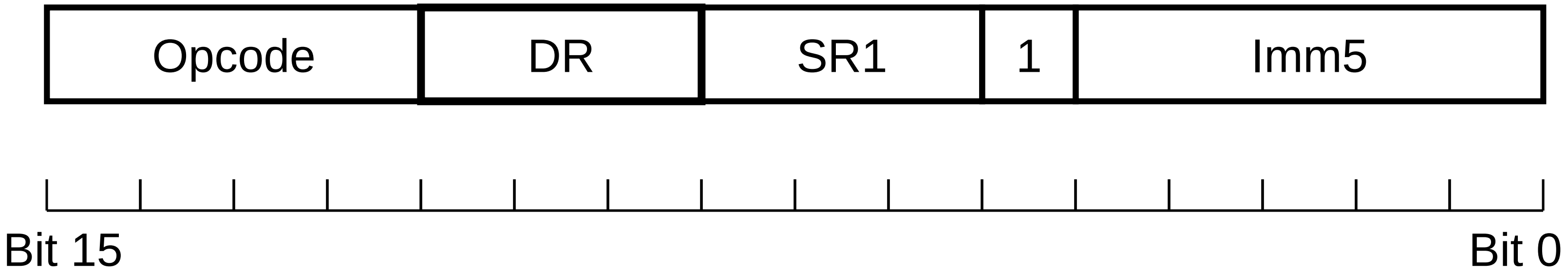
ILLINOIS

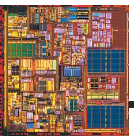
LC-3bInstruction Formats

- ADD, AND (without Immediate)



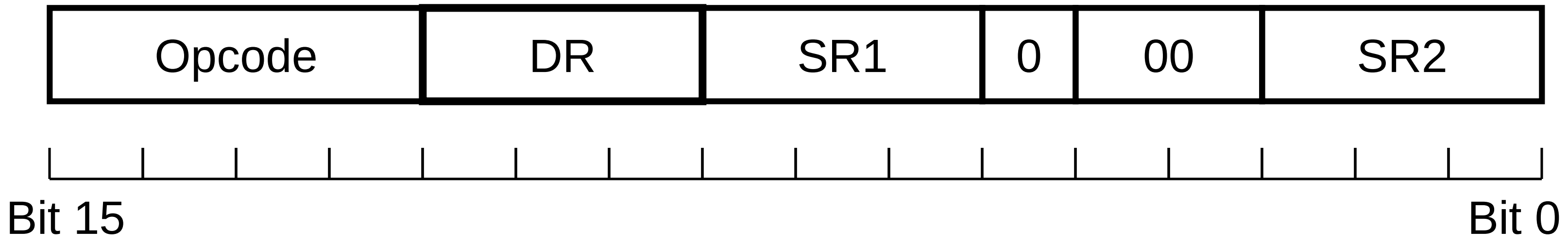
- ADD, AND (with Immediate), NOT



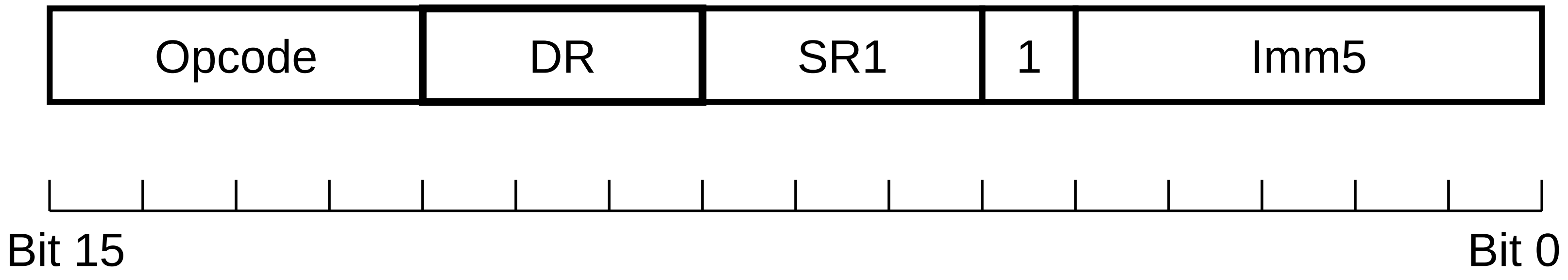


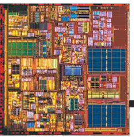
LC-3b Instruction Formats

- ADD, AND (without Immediate)



- ADD, AND (with Immediate), NOT





MIPS Instruction Formats

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
r format	**OP**	**rs**	**rt**	**rd**	**sa**	**funct**
i format	**OP**	**rs**	**rt**	**immediate**		
j format	**OP**	**target**				

- for instance, "`add r1, r2, r3`" has
  - OP=0, rs=2, rt=3, rd=1, sa=0 (shift amount), funct=32
  - 000000 00010 00011 00001 00000 100000
- opcode (OP) tells the machine which format

MIPS ISA Tradeoffs

6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
OP	**rs**	**rt**	**rd**	**sa**	**funct**
OP	**rs**	**rt**	**immediate**		
OP	**target**				

What if?

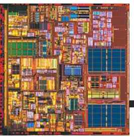
- 64 registers
- 20-bit immediates
- 4 operand instruction (e.g. Y = AX + B)

Think about how sparsely the bits are used

ILLINOIS

Conditional branch

- How do you specify the destination of a branch/jump?
  - Theoretically, the destination is a full address
    - 16 bits for LC3b
    - 32 bits for MIPS

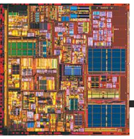


Conditional branch

- How do you specify the destination of a branch/jump?
- studies show that almost all conditional branches go short distances from the current program counter (loops, if-then-else).

Conditional branch

- How do you specify the destination of a branch/jump?
- studies show that almost all conditional branches go short distances from the current program counter (loops, if-then-else).
  - we can specify a relative address in much fewer bits than an absolute address
  - e.g., beq $1, $2, 100 => if ($1 == $2) PC = PC + 100 * 4



Conditional branch

- How do you specify the destination of a branch/jump?
- studies show that almost all conditional branches go short distances from the current program counter (loops, if-then-else).
  - we can specify a relative address in much fewer bits than an absolute address
  - e.g., beq $1, $2, 100   => if ($1 == $2) PC = PC + 100 * 4
- How do we specify the condition of the branch?

MIPS conditional branches

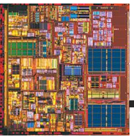
- beq, bne   *beq r1, r2, addr => if (r1 == r2) goto addr*
- slt $1, $2, $3 => if ($2 < $3) $1 = 1; else $1 = 0
- these, combined with $0, can implement all fundamental branch conditions

Always, never, !=, = =, >, <=, >=, <, >(unsigned), <= (unsigned), ...

```
if (i<j)
    w = w+1;
else
    w = 5;
```

⇒

```
slt $temp, $i, $j
beq $temp, $0, L1
add $w, $w, #1
beq $0, $0, L2
L1: add $w, $0, #5
L2:
```



Jumps

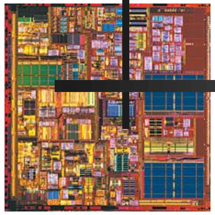
- need to be able to jump to an absolute address sometimes
- need to be able to do procedure calls and returns

- jump -- j 10000 => PC = 10000
- jump and link -- jal 100000 => $31 = PC + 4; PC = 10000
  - used for procedure calls

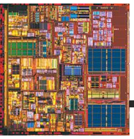
OP	**target**

- jump register -- jr $31 => PC = $31
  - used for returns, but can be useful for lots of other things.

Computer Performance



Computer Performance: TIME, TIME, TIME



- Response Time (latency)
  - — How long does it take to execute a job?
  - — How long must I wait for the database query?
- Throughput
  - — How many jobs can the machine complete in a minute?
  - — What is the average execution rate?
  - — How much work is getting done?

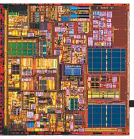
- *If we upgrade a machine with a new processor what do we improve?*
- *If we add a new machine to the lab what do we improve?*

ILLINOIS

A Throughput Oriented Approach

- Each car still experience the same latency at the toll
- Many cars are served to improve the number of cars processed per time unit (sec/min/hr)

Case Study



- How can we improve the throughput of I-57 between Chicago and Champaign?
  - Increase the number of lanes

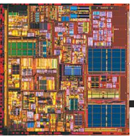
- How can we improve the latency of driving from Chicago to Champaign on I-57?
  - Increase the driving speed

ILLINOIS

Aspects of CPU Performance

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

	Inst Count	CPI	Clock Rate
Program	X		
Compiler	X	X	
Inst. Set.	X	X	
Organization	X	X	X
Technology			X



P&H Definition of Performance

- For some program running on machine X,

  $$\text{Performance}_X = 1 \,/\, \text{Execution time}_X$$

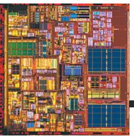
- "X is n times faster than Y"

  $$\text{Performance}_X \;/\; \text{Performance}_Y = n$$

- Problem:
  - machine A runs a program in 20 seconds
  - machine B runs the same program in 25 seconds
  - A is __1.25___ times faster than B?

ILLINOIS

How to Improve Performance



$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

So, to improve performance (everything else being equal) you can either

_________ the # of required cycles for a program, or

_________ the clock cycle time or, said another way,

_________ the clock rate.

Performance Releated Metrics

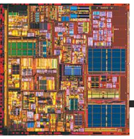
- An execution of a given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds
- We have a vocabulary of metrics that relate these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)

    a floating point intensive application might have a higher CPI
  - MIPS (millions of instructions per second)

    this would be higher for a program using simple instructions

Instruction Set Metrics Example

- *If two machines have the same ISA which of our quantities (e.g., clock rate, CPI, execution time, # of instructions, MIPS) will likely be identical during a comparison?*

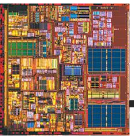


CPI Example

- Suppose we have two implementations of the same instruction set architecture (ISA).

For some program,
Machine A has a clock cycle time of 10 ns. and a CPI of 2.0
Machine B has a clock cycle time of 20 ns. and a CPI of 1.2

What machine is faster for this program, and by how much?



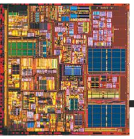
Cycles and CPI Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).

  The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
  The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
  - First: 2*1 + 1*2 + 2*3 = 10 cycles
  - Second: 4*1 + 1*2 + 1*3 = 9 cycles
  - Second is faster than first by 1 cycle or about 11%

- What is the average CPI for each sequence?
  - First: 10/5 = 2
  - Second: 9/6 = 1.5



MIPS example

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

- Which sequence will have higher MIPS rating?
  - First: (5M + 1M + 1M) / ((5M*1+1M*2+1M*3)/100M) = 70 MIPS
  - Second: (10M+1M+1M) / ((10M*1+1M*2+1M*3)/100M) = 80 MIPS

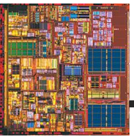
MIPS example (cont.)

- Two different compilers are being tested for a 100 MHz. machine with three different classes of instructions: Class A, Class B, and Class C, which require one, two, and three cycles (respectively). Both compilers are used to produce code for a large piece of software.

  The first compiler's code uses 5 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

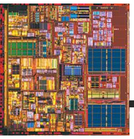
  The second compiler's code uses 10 million Class A instructions, 1 million Class B instructions, and 1 million Class C instructions.

- Which sequence will be faster according to execution time?
  - First: 5M*1+1M*2+1M*3 = 10M cycles
  - Second: 10M*1+1M*2+1M*3 = 15M cycles



MP1 - Important!

- All remaining MP due dates include an automatic 2-day extension.
  - Treat Fridays as the real deadline!
  - Extended due dates are on Sundays. Use the two days as contingency.
- Read the document carefully before you start!
- Review relevant parts of the document as you progress!
- Follow instructions and steps!



READ 1.6 AND CHAPTER 2 START MP1