CSE 31
Computer Organization

Lecture 20 – Instruction Format (cont)

Announcements

- Labs
  - Lab 7 grace period* ends this week
    - » Demo is REQUIRED to receive full credit
  - Lab 8 due this week (with **7 days grace period*** after due date)
    - » Demo is REQUIRED to receive full credit
  - Lab 9 out this week
    - » Due at 11:59pm on the same day of your lab after next (with 7 days grace period* after due date)
    - » You must demo your submission to your TA within 14 days from posting of lab
    - » Demo is REQUIRED to receive full credit
- Reading assignments
  - Reading 06 (zyBooks 6.4 – 6.7) due **tonight**, 10-APR and Reading 07 (zyBooks 5.1 – 5.6) due 17-APR
    - » Complete **Participation Activities** in each section to receive grade
    - » IMPORTANT: Make sure to submit score to CatCourses by using the link provided on CatCourses

* A 10% penalty will be applied for late *submissions* during the grace period.

Announcements

- Homework assignment
  - Homework 05 (zyBooks 1.6 – 1.7) due **tonight**, 10-APR and Homework 06 (zyBooks 5.1 – 5.11) due 01-MAY
    - » Complete **Challenge Activities** in each section to receive grade
    - » IMPORTANT: Make sure to submit score to CatCourses by using the link provided on CatCourses
- Midterm 02
  - On 19-APR during class
  - Announcement and Sample Exam posted by tonight
- Project 02
  - Due 05-MAY
  - Can work in teams of 2 students
    - » Each team member must identify teammate in "Comments…" text-box at the submission page
    - » If working in teams, each student must submit code (can be the same as teammate) and demo individually
    - » Grade can vary among teammates depending on demo
  - Demo required for project grade
    - » No partial credit for submission without demo
  - **No grace period**
    - » **Must complete submission and demo by due date.**

R-Format Instructions (review)

- Define "fields" of the following number of bits each: 6 + 5 + 5 + 5 + 5 + 6 = 32

6	5	5	5	5	6

- For simplicity, each field has a name:

opcode	rs	rt	rd	shamt	funct

- Meaning of field:
  - **`opcode`**: partially specifies what is the instruction (=0 for R-Format)
  - **`funct`**: combined with `opcode`, exactly specifies the instruction
  - **`rs`** (Source Register): generally specifies register containing **1st operand**
  - **`rt`** (Target Register): generally specifies register containing **2nd operand**
  - **`rd`** (Destination Register): generally specifies register that **receives result**

R-Format Example (1/2)

- MIPS Instruction:

```
add    $t0,$t1,$t2
add    $8,$9,$10
```

`opcode` = **0** (look up in table)

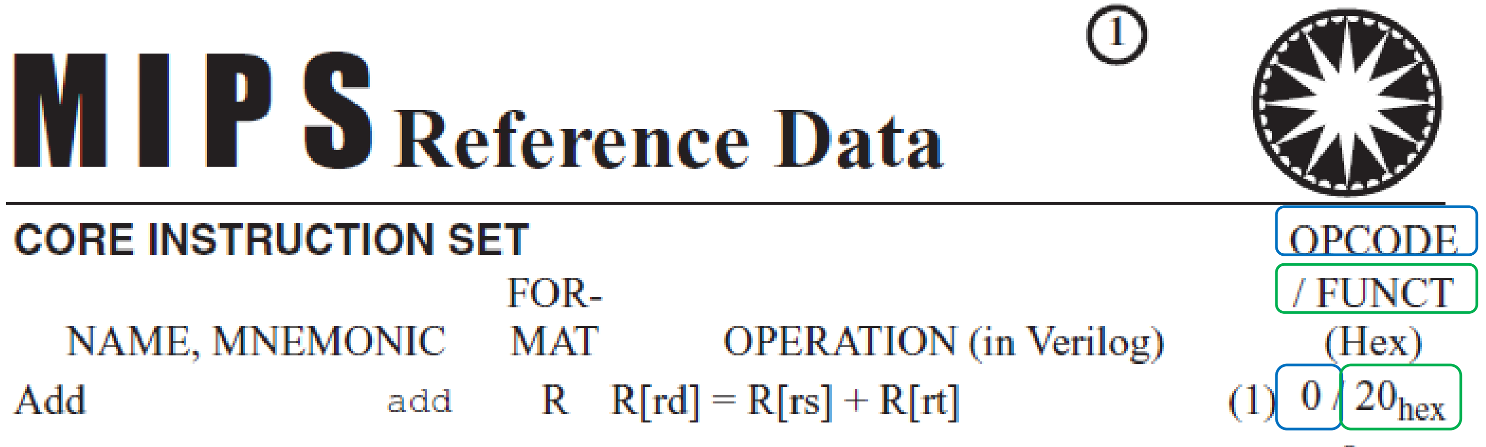
`funct` = **32** (look up in table)

`rd` = 8 (destination)

`rs` = 9 (first operand)

`rt` = 10 (second operand)

`shamt` = 0 (not a shift)



MIPS Reference Data (1)

CORE INSTRUCTION SET

NAME, MNEMONIC		FOR-MAT	OPERATION (in Verilog)		OPCODE / FUNCT (Hex)
Add	add	R	R[rd] = R[rs] + R[rt]	(1)	0 / 20_{hex}

R-Format Example (2/2)

- MIPS Instruction:

  `add    $8,$9,$10`

  Decimal number per field representation:

0	9	10	8	0	32

  Binary number per field representation:

000000	01001	01010	01000	00000	100000

hex

hex representation: `012A 4020`$_{hex}$

decimal representation: `19,546,144`$_{ten}$

Called a Machine Language Instruction

I-Format Instructions (1/4)

- What about instructions with immediates?
  - 5-bit field only represents numbers up to the value 31: immediates may be much larger than this
  - Ideally, MIPS would have only one instruction format (for simplicity): unfortunately, we need to compromise

- Define new instruction format that is partially consistent with R-format:
  - First notice that, if instruction has immediate, then it uses at most 2 registers.

I-Format Instructions (2/4)

- Define "fields" of the following number of bits each: 6 + 5 + 5 + 16 = 32 bits

6	5	5	16

  - Again, each field has a name:

opcode	rs	rt	immediate

  - Key Concept: Only one field is inconsistent with R-format. Most importantly, `opcode` is still in same location.

I-Format Instructions (3/4)

- What do these fields mean?
  - **`opcode`**: same as before except that, since there's no `funct` field, `opcode` uniquely specifies an instruction in I-format
    - » This also answers question of why R-format has two 6-bit fields to identify instruction instead of a single 12-bit field: in order to be consistent as possible with other formats while leaving as much space as possible for immediate field.
  - **`rs`**: specifies a register operand (if there is one)
  - **`rt`**: specifies register which will receive result of computation (this is why it's called the target register "`rt`") or other operand for some instructions.

I-Format Instructions (4/4)

- The Immediate Field:
  - **`addi`**, **`slti`**, **`sltiu`**, the immediate is **sign-extended** to 32 bits. Thus, it's treated as a signed integer.
  - 16 bits ➔ can be used to represent immediate up to 2^{16} different values
  - This is large enough to handle the offset in a typical **`lw`** or **`sw`**, plus a vast majority of values that will be used in the **`slti`** instruction.
  - We'll see what to do when the number is too big later

I-Format Example (1/2)

- MIPS Instruction:

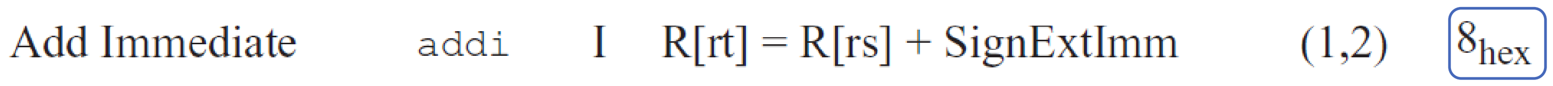
```
addi    $s5,$s6,-50
addi    $21,$22,-50
```

`opcode` = **8** (look up in table MIPS_Reference_Sheet.pdf)

`rs` = 22 (register containing operand)

`rt` = 21 (target register)

`immediate` = -50 (by default, this is decimal)



Add Immediate	`addi`	I	R[rt] = R[rs] + SignExtImm	(1,2)	$\boxed{8_{hex}}$

I-Format Example (2/2)

- MIPS Instruction:

```
addi   $21,$22,-50
```

Decimal/field representation:

8	22	21	-50

Binary/field representation:

001000	10110	10101	1111111111001110

hexadecimal representation: $\texttt{22D5 FFCE}_{\texttt{hex}}$

decimal representation: $\texttt{584,449,998}_{\texttt{ten}}$

Quiz

Which instruction has same representation as 35_{ten}?

a) add \$0, \$0, \$0
b) subu \$s0,\$s0,\$s0
c) lw \$0, 0(\$0)
d) addi \$0, \$0, 35
e) subu \$0, \$0, \$0

opcode	rs	rt	rd	shamt	funct
opcode	rs	rt	rd	shamt	funct
opcode	rs	rt	offset		
opcode	rs	rt	immediate		
opcode	rs	rt	rd	shamt	funct

Registers numbers and names:

0: \$0

8: \$t0, 9: \$t1, 10: \$t2, 11: \$t3, 12: \$t4, 13: \$t5, 14: \$t6, 15: \$t7

16: \$s0, 17: \$s1, 18: \$s2, 19: \$s3, 20: \$s4, 21: \$s5, 22: \$s6, 23: \$s7

Opcodes and function fields (if necessary)

add: opcode = 0, funct = 32

subu: opcode = 0, funct = 35

addi: opcode = 8

lw: opcode = 35

Quiz

Which instruction has same representation as 35_{ten}?

a) add \$0, \$0, \$0
b) subu \$s0,\$s0,\$s0
c) lw \$0, 0(\$0)
d) addi \$0, \$0, 35
e) subu \$0, \$0, \$0

opcode	rs	rt	rd	shamt	funct
opcode	rs	rt	rd	shamt	funct
opcode	rs	rt	offset		
opcode	rs	rt	immediate		
opcode	rs	rt	rd	shamt	funct

Registers numbers and names:

0: \$0
8: \$t0, 9: \$t1, 10: \$t2, 11: \$t3, 12: \$t4, 13: \$t5, 14: \$t6, 15: \$t7
16: \$s0, 17: \$s1, 18: \$s2, 19: \$s3, 20: \$s4, 21: \$s5, 22: \$s6, 23: \$s7

Opcodes and function fields (if necessary)

`add`: opcode = 0, funct = 32
`subu`: opcode = 0, funct = 35
`addi`: opcode = 8
`lw`: opcode = 35

I-Format Problem (1/3)

- Problem:
  - Chances are that `addi`, `lw`, `sw` and `slti` will use immediates small enough to fit in the immediate field.
  - …but what if it is too big?
    - » `addi $s0, $s1, 400000000`
    - » We need a way to deal with a 32-bit immediate in any I-format instruction.

I-Format Problem (2/3)

- Solution to Problem:
  - Handle it in software + new instruction
  - Don't change the current instructions: instead, add a new instruction to help out

- New instruction:

```
    lui    register, immediate
```

  - stands for ***Load Upper Immediate***
  - takes 16-bit immediate and puts these bits in the upper half (high order half) of the register
  - sets lower half to `0`s

I-Format Problems (3/3)

- Solution to Problem (continued):
  - So how does `lui` help us?
  - Example:

```
addiu $t0,$t0, 0xABABCDCD
```

…becomes

```
lui $at 0xABAB
ori $at, $at, 0xCDCD
addu $t0,$t0,$at
```

R-format!

  - Now each I-format instruction has only a 16-bit immediate.
  - Wouldn’t it be nice if the assembler would do this for us automatically? (later)