MIPS Instruction Set

D. Venkata Vara Prasad

Professor

SSN College of Engineering

Session Meta Data

Author	C.Annadurai
Version No	1.1
Release Date	14.03.2022
Reviewer	

Revision History

Date of Revision	Details	Version Number

Session Objectives

- To explain the MIPS Instruction Set of the Computer system.

Session Outcomes

- At the end of the session, students will be able to
  - Understand the concept of MIPS Instruction Set.

Outline

- To discuss about
    - ❖ MIP Instruction set.

- R format
- I format
- J format

MIPS Instruction Set

Advantages

- Typical of Modern RISC Instruction sets
- Free Simulator Available for Unix, PCs and Macs
- Used in real machines - MIPS R2000 Processors
- Used in many CS Compiler Classes as Target Machine

MIPS Architecture

- 32-bit RISC Processor
- 32 32-bit Registers, \$0..\$31
- Pipelined Execution of Instructions
- All instructions are 32-bits
- Most Instructions executed in one clock cycle
- 2^{30} 32-bit memory words
- Byte addressable memory
- A 32-bit word contains four bytes
- To address the next word of memory add 4

MIPS Instruction Formats

- R format - Uses three register operands
  - Used by all arithmetic and logical instructions
- I format - Uses two register operands and an address/immediate value
  - Used by load and store instructions
  - Used by arithmetic and logical instructions with a constant
- J format - Contains a jump address
  - Used by Jump instructions

MIPS Instruction Formats

32-bit Instruction Formats R, I and J

OP	RS	RT	RD	SHAMT	FUNCT

OP	RS	RT	Address/Immediate

OP	Jump Address

MIPS Instruction Formats

32-bit Instruction Formats R

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

op: Basic operation of the instruction, traditionally called the opcode.
rs: The first register source operand.
rt: The second register source operand.
rd: The register destination operand. It gets the result of the operation.
shamt: Shift amount. (explains shift instructions and this term; it will not be used until then, and hence the field contains zero in this section.)
funct: Function. This field, often called the function code, selects the specific variant of the operation in the op field.

MIPS Instruction Formats

32-bit Instruction Formats I

op	rs	rt	constant or address
6 bits	5 bits	5 bits	16 bits

A second type of instruction format is called I-type (for immediate) or I-format and is used by the immediate and data transfer instructions.

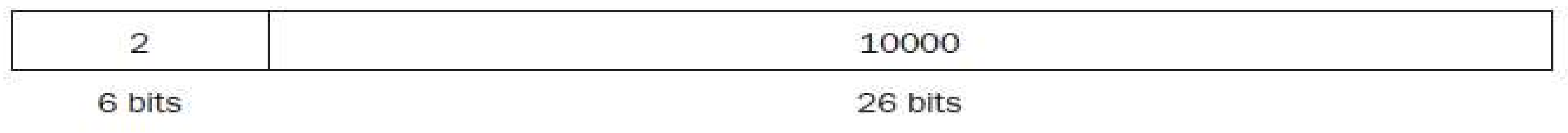
The 16-bit address means a load word instruction can load any word within a region of 32,768 bytes (8192 words) of the address in the base register rs. Similarly, add immediate is limited to constants no larger than $\pm 2^{15}$.

We see that more than 32 registers would be difficult in this format, as the rs and rt fields would each need another bit, making it harder to fit everything in one word.

MIPS Instruction Formats

32-bit Instruction Formats J

OP	Jump Address



The final MIPS instruction format, called the J-type, which consists of 6 bits for the operation field and the rest of the bits for the address field

j   10000 # go to location 10000
where the value of the jump opcode is 2 and
 the jump address is 10000.

MIPS Instruction Set

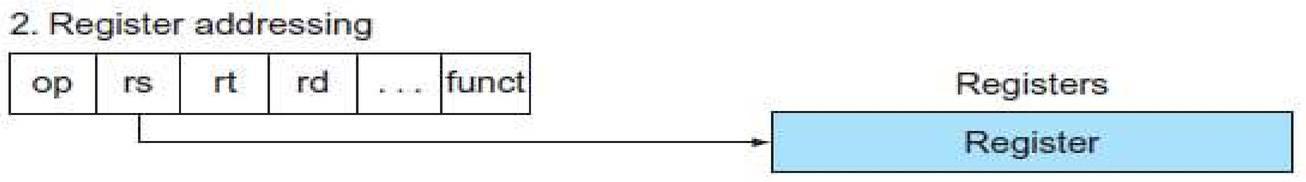
Only Load instruction can read an operand from memory
Only Store instruction can write an operand to memory

Typical RISC calculations require

- Load(s) to get operands from memory into registers
- Calculations performed only on values in registers
- Store(s) to write result from register back to memory

MIPS Addressing Modes

- Register - Uses value in register as operand
  Example $2 - Uses register 2 as operand
- Direct Address - Uses value stored in memory at given address
  Example 100 - Uses value stored at location 100 in memory as operand
- Register Indirect Address - Uses value in register as address of operand in memory
  Example ($3) - Uses value in register 3 as address of memory operand

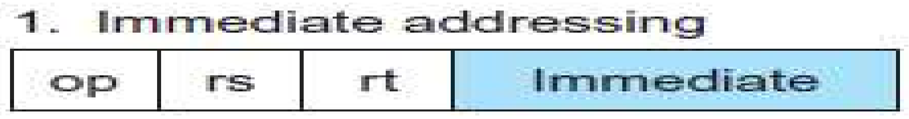


MIPS Addressing Modes

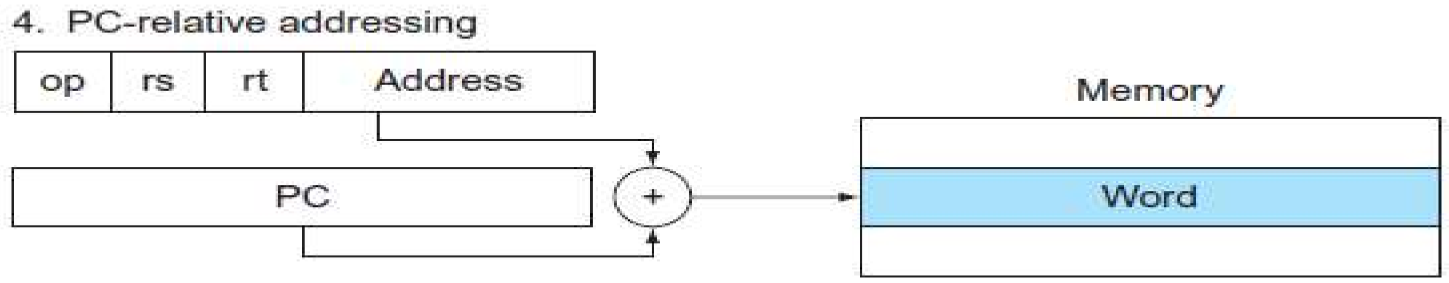
- Indexed Address - Adds address field and register value and uses this as address of operand in memory
  - Example 100($2) - Adds 100 to the value in register 2 and reads the operand from the resulting memory address
  - Used for array X[I] operations, the array index is normally in the register and the address field is the first location in the array

MIPS Addressing Modes

- Immediate Addressing - Uses constant value contained in instruction
  Example addi $1,$2,4 - adds constant 4 to register 2 and stores result in register 1
  Used for constants in programs



- PC relative - Address from instruction is added to the current value in the Program Counter
  Example J 4 - Jumps to address PC+4
  Saves address bits in jump instructions



MIPS Assembly Language Examples

- ADD \$1,\$2,\$3
  Register 1 = Register 2 + Register 3
- SUB \$1,\$2,\$3
  Register 1 = Register 2 - Register 3
- AND \$1,\$2,\$3
  Register 1 = Register 2 AND Register 3
- ADDI \$1,\$2,10
  Register 1 = Register 2 + 10
- SLL \$1, \$2, 10
  Register 1 = Register 2 shifted left 10 bits

MIPS Assembly Language Examples

- LW $1,100
  Register 1 = Contents of memory location 100
  Used to load registers
- SW $1,100
  Contents of memory location 100 = Register 1
  Used to save registers
- LUI $1,100
  Register 1 upper 16-bits = 100
  Lower 16-bits are set to all 0's
  Used to load a constant in the upper 16-bits
  Other constants in instructions are only 16-bits, so this instruction is used when a constant larger than 16-bits is required for the operation

MIPS Assembly Language Examples

- J 100
  Jumps to PC+100
- JAL 100
  Save PC in $31 and Jumps to PC+100
  Used for subroutine calls
- JR $31
  Jumps to address in register 31
  Used for subroutine returns

MIPS Assembly Language Examples

- BEQ $1, $2, 100
  If register 1 equal to register 2 jump to PC+100
  Used for Assembly Language If statements
- BNE $1, $2, 100
  If register 1 not equal to register 2 jump to PC+100
  Used for Assembly Language If statements

MIPS Assembly Language Examples

- SLT $1,$2,$3
  If register 2 is less than register 3 then register 1=1 else register 1=0
  In an assembly language flag a “1” means true and a “0” means false
  Used in conjunction with **Bxx** instruction to implement any arithmetic comparison
  Required for more complex assembly language If statements

MIPS Program Examples
A = B + C;

```
LW  $2, B              ;Register 2 = value of B
LW  $3, C              ;Register 3 = value of C
ADD $4, $2, $3         ;Register 4 = B+C
SW  $4, A              ; A = B + C
```

MIPS Assembly Language Label Examples

```
N:      WORD        0
```

Like declaring an Integer, N, in a HLL

Sets up N as a Label that points to a 32-bit data value

Initial value is set to 0

```
LOOP:ADD $a0,$a0,$a1
     J LOOP
```

Sets up LOOP as a Label that points to the Add instruction

Can jump to LOOP.

MIPS Assembler Directives

- Assembler directives are commands for the assembler that do not generate machine instructions
- Assembler directives are also called pseudo ops Used to set up data and instruction areas

MIPS Assembler Directive Examples

.DATA

The following lines are constant or data values

.WORD

Reserve a 32-bit word in memory for a variable

.ASCII

Place a string in memory using the ASCII character code

.TEXT

The following lines are instructions

MIPS Examples

```
        # f=(g+h)-(i+j);
     #  compiler  puts f,g,h,i in
    #       $16,$17,$18,$19,$20
add $8,$17,$18               # $8 =g+h
add $9,$19,$20                 #9 =i+j
sub $16,$8,$9   # $16=(g+h)-(i+j)
```

MIPS Examples

```
# g=h+A[i]
# compiler puts g,h,i in $17,$18,$19
LW $8,Astart($19)  # $8 = A[i]
ADD $17,$18,$8  #  $17 = h+A[i]
```

MIPS Examples

```
# A[i] = h + A[i];
# compiler puts g,h in $17,$18
# compiler puts i times 4 in $19
LW $8,Astart($19)   # $8 = A[i]
ADD $8,$18,$8    # $8  =  h+A[i]
SW $8,Astart($19)   # A[i]=h+A[i]
```

MIPS Examples

```
#if (i==j) f=g+h;

      Bne $19,$20,Endif
      Add $16,$17,$18
Endif:
```

MIPS Examples

```
#if (i==j) f=g+h; else f=g-h;

       Bne $19,$20,Else
       Add $16,$17,$18
       J Exit
Else:  sub $16,$17,$18
Exit:
```

MIPS Examples

#for loop

```
       ADDI    $5,$0,10
Loop:  …
       …
            …
            ADDI    $5,$5,-1
            BNE     $5,$0,Loop
```

Logical Instructions

and, or, xor, nor, andi (and immediate), ori (or immediate), sll (shift left logical), srl (shift right logical), sra (shift right arithmetic),

examples

```
sll $1, $2, 10 ; $1 = $2 << 10 (shift left logical 10
bits of $2)
srl $1, $2, 10 ; $1 = $2 >> 10 (shift right logical
10 bit)
or $1, $2, $3 ; $1 = $2 | $3
and $1, $2, $3 ; $1 = $2 & $3
ori $2, $3, 99 : $2 = $3 | 99
```

Summary

- The MIPS Instruction Set of the Computer system were discussed in detail.

References

- David A. Patterson and John L. Hennessey, “Computer Organization and Design‟, Fifth edition, Morgan Kauffman / Elsevier, 2014.
- V.Carl Hamacher, Zvonko G. Varanesic and Safat G. Zaky, “Computer Organisation“, VI edition, Mc Graw-Hill Inc, 2012.
- William Stallings “Computer Organization and Architecture”, Seventh Edition , Pearson Education, 2006.
- Vincent P. Heuring, Harry F. Jordan, “Computer System Architecture”, Second Edition, Pearson Education, 2005.

Thank you