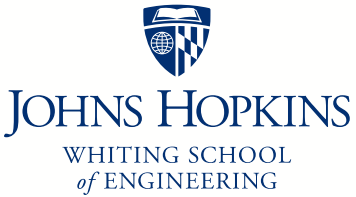
Johns Hopkins Engineering

Module 4: MIPS Control Structures

EN605.204: Computer Organization



Base/Base-offset Addressing

opcode (6)	rs (5)	rt (5)	immediate (16)

- Address of second operand is found at $rs (base) + offset
- **lw $t0, $t1(16)** -> $t0 = address inside of $t1 + 16 bytes
  - 'lw' is a "load" instruction
    - moves a value from memory into a register
  - Immediate value is 16-bit two's complement
  - Ex: if $t1 contains the address 0x0010, the instruction above adds the value in $t0 to the value at 0x0020

'lw' Using Base-offset Addressing

MIPS Code	Binary/Hex	Memory Locations
	00	**0x0013**
	00	**0x0012**
	00	**0x0011**
$t0 = 0x03	03	**0x0010**
	……	
lw $t0, $t1(16)	0x8D	**0x0003**
$t1 = 0x0000	0x28	**0x0002**
	0x00	**0x0001**
	0x10	**0x0000**

PC-relative Addressing

opcode (6)	rs (5)	rt (5)	immediate (16)

- Offset in immediate field is offset of Program Counter (PC)
  - beq $t0, $zero, 0xFFFC -> go to PC-4 if $t0 = 0
    - 'beq' is a conditional branch (if equal)Branch target is "near" the PC
    - Within 2^15 – 1 bytes to be exact
    - Can be used to implement "if", for/while loops

PC-relative Addressing

Let's assume $t0 = 5, how many times does this loop repeat?

PC				
0x00000000	**loop: addi**	**$t0**	**$t0**	**0xFFFF**
…	**…**	**…**	**…**	**…**
0x00000008	**bne**	**$t0**	**$zero**	**0xFFF4 (loop)**
0x0000000C	**…**	**…**	**…**	**…**

Why 0xFFF4? PC points to next instruction!

(Pseudo-) Direct Addressing

opcode (6)	offset (26)

- Address resides in the 26-bit offset field: **j 0x1234**
  - 0x1234 = 00 0000 0000 0001 0002 0003 0004 (26 bits)
  - MIPS programs loaded into 1 of 16 256MB "blocks"
    - 16 blocks: leading 4 bits go from [0000], [0001], … , [1111]
    - Leading 4 bits of PC will always be one of these blocks
  - If offset is only 26 bits, how do we address a block with 2^32 addresses?
  - Shift left 2 bits: 00 0000 0000 0100 0200 0300 0400
    - Offset in words, but we need bytes (same as multiplying by 4)
  - Fill in top 4 bits with 4 MSB's (block address) from PC
  - This gives us a full 32-bits of accessible memory locations!

Read: https://www.eecs.yorku.ca/course_archive/2004-05/F/2021/notes/UnderstandingTheJumpInstruction.pdf

Conditional Operators

Conditional Operator	MIPS Instruction
x == y	beq $t0, $t1, <LABEL>
x != y	bne $t0, $t1, <LABEL>
x < y	blt $t0, $t1, <LABEL>
x > y	bgt $t0, $t1, <LABEL>
x <= y	ble $t0, $t1, <LABEL>
x >= y	bge $t0, $t1, <LABEL>
x == 0	beqz $t0, <LABEL>

Control Structures

'for' loop	'while' Loop

`# for i from 0 to 5`
`addi $t0, $zero, 5`
`addi $t1, $zero, 0`
 	
`loop:`
`beq $t1, $t0, end`
`...`
`addi $t1, $t1, 1`
`j loop`
 	
`end:`
`...`
	`# loop while $t0 <= 10`
	`while:`
	`...`
	`bgt $t0, 10, end`
	
	`j while`
	
	`end:`
	`...`