MIPS Instructions

add a, b, c	R	is	a = b + c]	
addi, a, b, #	I	is	a = b + #	*# = [-, +]
sub, a, b, c	R	is	a = b - c	
mul a, b, c	R	is	a = b * c	
and, a, b, c		is	a = b && c	
andi, a, b, #	I	is	a = b && #	
or, a, b, c		is	a = b v c	
ori, a, b, #	I	is	a = b v #	
nor				
lw, a, #(b)	I	is	load b[#/4] into a	
sw, ax, #(b)	I	is	save a into b[#/4]	
lb, a, #(b)	I	is	load b + # into a	
lbu, a, #(b)				
sb, a, #(b)	I	is	save a into b + #	
lui, a, #	I	is	a = # * 2^16	
beq, a, b, L	I	is	if (a == b) go to L	
bne, a, b, L	I	is	if (a != b) go to L	
blt a, b, L	I	is	if (a < b) go to L	
beg a, b, L	I	is	if (a => b) go to L	
slt, a, b, c	R	is	if (b < c) a = 1 else a = 0	
slti, a, b, #	I	is	if (b < #) a = 1 else a = 0	
sltu, a, b, c		is	if (b < c) a = 1 else a = 0 for unsigned ints	
sltui, a, b, #	I	is	if (b < #) a = 1 else a = 0 for unsigned ints	
jump L	J	is	jump to L	
sll, a, b, #	R	is	a = b * 2^#	
srl,a , b, #	R	is	a = b / 2^#	
jal L	J			
jr $ra	J			
ll				
sc				
move a, b		is	a = b	

Lecture 4.1: MIPS ISA: Introduction Lecture 4.1: MIPS ISA: Introduction

- MIPS has 32 x 32-bit registers which is 32 x 4-byte registers
- Assembler names Assembler names
  - t registers are for temporary values
  - s registers are for saved values
- Memory is byte addressed
  - Load and Save commands load and save register values from and into memory
  - Words are aligned in memory

- Design principles
    - 1: simplicity favours regularity
    - 2: smaller is faster
    - 3: make common case faster
    - 4: good design requires good compromises

- MIPS is Big Endian: most significant byte at the least address
- Register $zero cannot be overwritten
- There are 3 different MIPS instruction formats
    - R: OP-6, rs-5, rt-5, rd-5, shamt-5, funct-6
    - I: OP-6, rs-5, rt-5, immediate-16
    - J: OP, jump target
        - OP = operation code
        - rs = first source register number
        - rt = second source register number
        - rd = destination register number
        - shamt = shift amount
        - funct = function code
- MIPS registers

$zero	0	constant zero
$at	1	
$v0-$v1	2-3	return values
$a0-$a3	4-7	arguments
$t0-$t7	8-15	temps
$s0-$s7	16-23	saved values
$t8-$t9	24-25	temps
$k	26-27	TRAP
$gp	28	global pointer
$sp	29	stack pointer
$fp	30	frame pointer
$ra	31	return address

Lecture 4.2: MIPS ISA -- Instruction Representation

- Basic block is a block of instructions that does not contains any branches or labels.
- beq and bne is faster than bge and blt
- Operand addressing modes
    - register: value is in the register
    - base: value is in memory
        - register relative: relative the register value
        - pseudo direct: the register value
    - immediate: value is a constant

- Instruction addressing modes
    - PC-relative: conditional branching
    - Pseudo-direct: jumps

Lecture 4.3: MIPS ISA -- Procedures, Compilation

- Procedure calling
    - place parameters into registers
    - transfer control to procedure
    - acquire storage for procedure
    - perform procedures operation
    - place result in register for caller
    - return to place of call
- procedure call: jal
- procedure return: jr
- memory layout
    - Text: program code
    - Static data: global variables
    - dynamic data: heap
    - stack: storage