Notes on Processor Instruction Set

The instruction set is is similar to a MIPS 2000 processor. There are 32 general purpose registers r0, …, r31, each 32-bit wide. Instructions come in 3 formats: R-type, I-type or J-type as shown in Table 1. Table 2 shows the 11 instructions we will implement.

Instructions are 32-bit wide. The break-up of the 32 bits into different fields depends on the instruction type as shown in Table 1.

R Type:

opcode:	always 000000
shamt:	000000 (for all the instructions we will implement)
funct:	6-bit field denotes the actual instruction as shown in Table 2.
rs, rt, rd:	register identifiers; uses 5 bits to encode a register number 0 through 31.

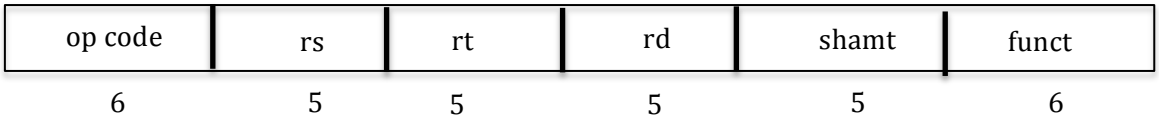
I Type:

opcode:	the value of 6-bit field for our instructions is shown in Table 2.
rs, rt:	register identifiers; uses 5 bits to encode a register number 0 through 31.
immediate:	a 16-bit value (encodes immediate constants or address offsets)

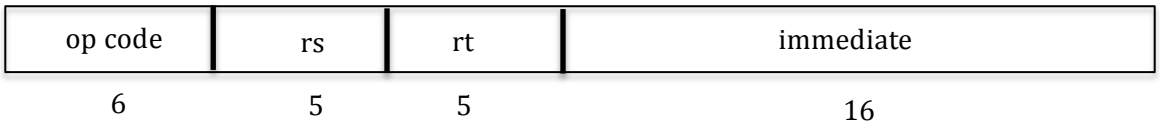
J Type:

opcode:	the value of 6-bit field for our instruction is shown in Table 2.
address:	a 26-bit field that encodes the 28 lower bits of the address of a jump.

R-Type



I-Type



J-Type

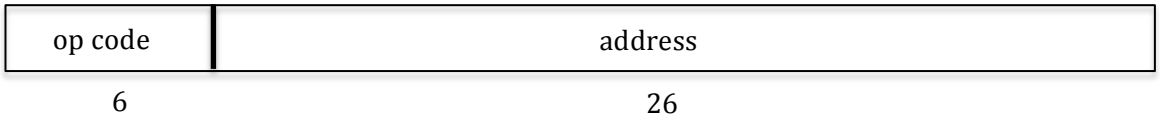


Table 1: Instruction Formats

Instruction	Type	opcode (hex)	funct (hex)	ALUop
add	R	0	20	1
and	R	0	24	3
or	R	0	25	4
slt	R	0	2A	5
sub	R	0	22	2
nop	R	0	0	-
addi	I	8	-	1
lw	I	20	-	1
sw	I	30	-	1
beq	I	4	-	6
jmp	J	2	-	-

Table 2: Instructions and their encoding

Instruction	Operation	Example	M/c Code
add rd, rs, rt	[rd] = [rs] + [rt]	add r1, r2, r3	00430820
and rd, rs, rt	[rd] = [rs] &[rt]	and r5, r6, r7	00C72824
or rd, rs, rt	[rd] = [rs] | [rt]	or r7, r8, r9	01093825
slt rd, rs, rt	*1	slt r9, r10, r11	014B482A
sub rd, rs, rt	rd] = [rs] - [rt]	sub r3, r4, r5	00851822
nop		nop	00000000
addi rt, rs, d	[rt] = [rs] +EXT(d)	addi r13, r14, -3	21CDFFFD
lw rt, d(rs)	*2	lw r11, -1(r12)	818BFFFF
sw rt, d(rs)	*3	sw r12, -2(r13)	C1ACFFFE
beq rs, rt, d	*4	beq r14, r15, -4	11CFFFFC
j -4	*5	j -4	0BFFFFFB

Notes on execution semantics.

*1: slt rd, rs, rt

if ([rs] < [rt]) [rd] = 1; else [rd] = 0; // does unsigned comparison

*2: lw rt, d(rs)

The immediate constant d (16 bits) is sign-extended to 32 bits and added to the contents of the base register rs , to form the 32-bit memory address for the load. The value read from memory is written to the destination register rt.

*3: sw rt, d(rs)

The immediate constant d (16 bits) is sign-extended to 32 bits and added to the contents of the base register rs , to form the 32-bit memory address for the store. The value read from source register rt is written to this memory location.

*4: beq rs, rt, d

The target address is computed by: (i) sign-extending the 16-bit offset d to 32-bits; (ii) shifting it left by two bit positions to scale by 4; (iii) adding it to the program counter pc. The branch condition is evaluated by comparing the contents of the two source registers rs and rt and setting the output to 1 or 0 depending on whether they are equal or not.

*5: j disp

The target address is computed by (i) shifting the 26 bit displacement d left by two bit positions to scale by 4; (ii) concatenating the 4 MSBs of the program counter pc with the 28 bit shifted displacement.