2.3

sub $s2, $s3, $s4

sll $s2, $s2, 2

add $s6, $s6, $s2

lw $s1, 0($s6)

sw $s1, 32($s7)

2.4

b[g] = a[f+1] + a[f];

f = a[f];

2.5

sll $t0, $s0, 2

add $t0, $s6, $t0

sll $t1, $s1, 2

add $t1, $s7, $t1

lw $s0, 0($t0)

lw $t0, 4($t0)

add $t0, $t0, $s0

sw $t0, 0($t1)

2.16

R-type

sub $v1, $v1, $v0

0000 0000 0110 0010 0001 1000 0010 0010

2.17

I-type

lw $v0, 4($at)

1000 1100 0010 0010 0000 0000 0000 0100

2.18

2.18.1

For R-type instructions you would need at least 7 bits for the rs, rt, and rd fields. The op code field could stay the same. The function field would probably also need to be extended to 7.

2.18.2

For I-type instructions you would need to extend the opcode field to 8 bits and rs, rt, and rd to 7 bits.

2.18.3

You could potentially get more done with less instructions, like being able to add 3 numbers in one instruction instead of 2 instructions if they add more register fields. The size of a word would probably increase with the proposed changes.

2.19

2.19.1

$t0 = 1010 1010 1010 1010 1010 1010 1010 1010 = 0xAAAAAAAA

sll $t2, $t0, 4

$t2 = 1010 1010 1010 1010 1010 1010 1010 0000 = 0xAAAAAAA0

or $t2, $t2, $t1

1010 1010 1010 1010 1010 1010 1010 0000

or 0001 0010 0011 0100 0101 0110 0111 1000

$t2= 1011 1010 1011 1110 1111 1110 1111 1000 = 0xBABEFEF8

2.19.2

sll $t2, $t0, 4

$t2 = 1010 1010 1010 1010 1010 1010 1010 0000 = 0xAAAAAAA0

andi $t2, $t2, −1

1010 1010 1010 1010 1010 1010 1010 0000

and 1111 1111 1111 1111 1111 1111 1111 1111

=1010 1010 1010 1010 1010 1010 1010 0000 = 0xAAAAAAA0

2.19.3

srl $t2, $t0, 3

srl 1010 1010 1010 1010 1010 1010 1010 1010

$t2= 0001 0101 0101 0101 0101 0101 0101 0101

andi $t2, $t2, 0xFFEF

0001 0101 0101 0101 0101 0101 0101 0101

and 0000 0000 0000 0000 1111 1111 1110 1111

0000 0000 0000 0000 0101 0101 0100 0101 = 0x00005545

2.20

andi $t0, $t0, 0x1f800

sll $t0, $t0, 15

andi $t1, $t1, 0x3ffffff

add $t1, $t1, $t0

2.23

slt $t2, $0, $t0 #$t2 = 1

bne $t2, $0, ELSE

j DONE

ELSE: addi $t2, $t2, 2 #t2 = 1+2 = 3

DONE:

$t2= 3

2.24

With the j instruction it is possible to reach 0x40000000 since the j instructions jumps straight to a specific address regardless of the current address. With the beq instruction it wouldn’t be possible to reach that address from 0x20000000 since beq jumps from the current address to an offset of that address. The offset has to be representable by 2^16 as beq only has 16 bits to represent the offset.

2.26

2.26.1

$t1 decrements by 1 each time so it should loop ten times. Since $t2 increases by 2 each loop $t2 should be 20 at the exit of the loop.

2.26.2

for(int i =10; i>0; i--)

{

b+=2;

}

2.26.3

If $t1 is initialized to N then the loop goes through N iterations. Since the loop executes 5 instructions per iteration then it would execute 5N instructions except it has to execute the first two instructions of the loop again to exit the loop, therefore it executes 5N + 2 instructions.