Page 432 Exercises 2.1, 2.2, 2.3, 2.6, 2.7, 2.8, 2.12, 2.19

2.1: For the following C statement, what is the corresponding MIPS assembly code? Assume that the C variables f, g, and h, have already been placed in registers $s0, $s1, and $s2, respectively. Use a minimal number of MIPS assembly instructions.

f= g + (h-5).

```
addi i, h, -5

add f, g, i
```

2.2: [5] <§2.2> Write a single C statement that corresponds to the

f = g + (h – 5); two MIPS assembly instructions below.

add f, g, h

add f, i, f

```
F = g + h

F = F+ i
```

2.3 [5] <§§2.2, 2.3> For the following C statement, write the corresponding MIPS assembly code. Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively. B[8] = A[i–j];

MIPS Code:

```
sub $t0, $s3, $s4       #load i - j into t0
```

```
sll $t0, $t0, 2         #shift left 2 to *4 to adjust to register
location.
add $t0, $t0, $s6       #add t0 to &A
lw $t0, 0[$t0]          #load value A[i-j] - into t0

sw $t0, 32[$s7]         #store t0 into B[8]
```

2.6 [5] <§2.4> Translate 0xabcdef12 into decimal.

a b c d e f 1 2

HEX

= 10 11 12 13 14 15 1 2 HEX

= 1010 1011 1100 1101 1110 1111 0001 0010

BINARY

= 2^(31)+2^(29)+2^(27)+2^(25)+2^(24)+2^(23)+2^(22)+2^(19)+2^(18)+2^(16)+2^(14)+2^(15)+2^(13)+2^(11)+2^(10)+2^(9)+2^(8)+2^(4)+2

DECIMAL

= 2 882 400 018

DECIMAL

2.7 [5] <§§2.2, 2.3> Translate the following C code to MIPS. Assume that the variables f, g, h, i, and j are

assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively. Assume that the elements of the arrays A and B are 8-byte words:

B[8] = A[i] + A[j];

```
sll $t0, $s3, 2      # $t0 = 4*i, as its 4 byte word
sll $t1, $s4, 2      # $t1 = 4*j, as its 4 byte word
add $t0, $t0, $s6    # A address is in s6 so A[i]
add $t1, $t1, $s6    # A address is in s6 so A[j]
lw $t0, 0($t0)       # Load A[i] into , $t0 = A[i]
lw $t1, 0($t1).      # Load A[j] into , $t1= A[j]
add $t0, $t1, $t0    # $t0=A[i]+A[j]
addi $t1, $s7, 32    # 4*8 is 32 as its 4 bytes word i.e and s7 has B, so t1 has
address of B[8]
sw $t0, 0($t1)       # B[8]=A[i]+A[j]
```

2.8 [10] <§§2.2, 2.3> Translate the following MIPS code to C.

Assume that the variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

```
addi $t0, $s6, 4    # $t0 = &A[1]
add $t1, $s6, $0    # $t1 = &A[0]
sw $t1, 0($t0)      # A[1] = &A[0]
lw $t0, 0($t0)      # $t0 = &A[0]
add $s0, $t1, $t0   # f = 2 * &A[0]
```

```
Answer: f = 2 * &A[0]
```

2.12 [5] <§§2.4, 2.5> Provide the type and assembly language instruction for the following binary value:0000 0010 0001 0000 1000 0000 0010 0000

```
00 0000 10000 10000 10000 00000 100000

Type: Add instruction (R-Type)

Instructions:

Add      6bits    100000

Shift    5bits    00000

Rd       5bits    10000    $s0

Rt       5bits    10000    $s0

Rs       5bits    10000    $s0

Therefore the instruction is: Add $s0 $s0 $s0
```

2.19 [5] <§2.6> Provide a minimal set of MIPS instructions that may be used to implement the following pseudoinstruction:

not $t1, $t2 // bit-wise invert

```
nor $t1,$t1,$t2
```