EE 352 Homework 1
Spring 2010 Nazarian

Name: ________________________ **Score:______**

Assigned: Friday, January 22

Due: Tuesday, February 2 at 9:30am (in class)

Pseudo-Instructions

1) (11 pts.) Expand/convert each of the following pseudo-instructions to an actual MIPS instruction sequence (you may use Section B.10 of the textbook for a list of instructions. Again you can't use the ones marked "pseudo-instructions".) For certain pseudo-instructions you will need to use the $at ($1) register as a temporary value (in fact, we have started this for you).

Note: For blanks, enter the register mnemonic like $t0, $at, etc. For immediate values write them as a 4-digit hex value (e.g. 0x0000).

Pseudo-Instruction	Actual Instruction Seq. 1
a) li $t0,0x7fffabcd	lui $at,_______ ori ____,_____,_____
b) clear $t0 (i.e. assign $t0=0)	Which of the following is not correct? a) XOR $t0,$t0,$t0 b) SUB $t0,$t0,$t0 c) ADD $t0,$0,$0 d) None of the above (i.e. they all work)
c) neg $t5,$t6 (i.e. $t5 = -$t6)	sub ____,_____,_____
d) bgt $t1,$t2,L1	slt $at,_____,_____ beq/bne $at,$0,L1 (select one)

Loads and Stores

2) (18 pts.) Examine each of the following instructions. For the load instructions show the entire 32-bit content of $t1 (in hex) after execution of the instruction.
Note: Show the full 32-bit hex value of $t1 after each load instructions. Precede the value with the hex modified ‘0x’ (e.g. 0x12345678).

$t0=	000071A8
$t1=	AB125680

ABCD EF01	M[0x71B8]
2345 6789	M[0x71B4]
BCDE F012	M[0x71B0]
3456 789A	M[0x71AC]
CDEF 0123	M[0x71A8]
4567 89AB	M[0x71A4]
DEF0 1234	M[0x71A0]

Instruction Result

a) `lw $t1,8($t0)` `$t1 =` ________________

b) `lbu $t1,0x13($t0)` `$t1 =` ________________

c) `lh $t1,-4($t0)` `$t1 =` ________________

d) `lw $t1,0xfffc($t0)` `$t1 =` ________________

e) `lb $t1,9($t0)` `$t1 =` ________________

f) `lhu $t1,-6($t0)` `$t1 =` ________________

g) `lb $t1,0xf($t0)` `$t1 =` ________________

h) `lhu $t1,-8($t0)` `$t1 =` ________________

i) `lb $t1,18($t0)` `$t1 =` ________________

3) **(7 pts.) Assume the contents of all memory locations shown below are initially 0.** Perform each store instruction and then show the contents of all memory locations after all stores have been executed.

Note: Show the full 32-bit hex value of each word of memory. The hex value should be preceded by hex modified ‘0x’ (e.g. 0x12345678). Remember memory locations not changed by the store instructions should be filled with 0’s.

$t0=	000071A8
$t1=	AB125680

Final Memory Contents After All Stores

	M[0x71B8]
	M[0x71B4]
	M[0x71B0]
	M[0x71AC]
	M[0x71A8]
	M[0x71A4]
	M[0x71A0]

		Instruction	Result
a)	sw	$t1,8($t0)	(Show in memory above)
b)	sb	$t1,0x0d($t0)	
c)	sh	$t1,-2($t0)	
d)	sb	$t1,0xfffa($t0)	
e)	sh	$t1,4($t0)	

MIPS Assembly

4) (26 pts.) The following assembly instruction sequence implements an expression of a C-style high level language.

a) Fill in the indicated result of each instruction. Assume the following initial values are true at the beginning of execution of the first instruction. Further assume all registers contain 0's initially.

Note: Show the full 32-bit hex value of each register after execution of the instruction. Begin the value with the hex modified '0x' (e.g. 0x12345678).

C Variable Name	Memory Contents	Start Address
Z	7E12 048A	0x1000E010
D	0000 0008	0x1000E00c
C	4FFD 3447	0x1000E008
B	0000 0005	0x1000E004
A	68CE 8932	0x1000E000

```
   i)   lui    $s0,0x1000      $s0 = ______________________
  ii)   ori    $s0,$s0,0xe000  $s0 = ______________________
 iii)   lw     $s1,0($s0)      $s1 = ______________________
  iv)   lw     $s2,4($s0)      $s2 = ______________________
   v)   lw     $s3,8($s0)      $s3 = ______________________
  vi)   addi   $s0,$s0,16      $s0 = ______________________
 vii)   lw     $s4,-4($s0)     $s4 = ______________________
viii)   mul    $t0,$s2,$s4     $t0 = ______________________
  ix)   addi   $t0,$t0,8       $t0 = ______________________
   x)   sub    $s1,$s1,$s3     $s1 = ______________________
  xi)   sll    $s1,$s1,1       $s1 = ______________________
 xii)   add    $s5,$t0,$s1     $s5 = ______________________
xiii)   sw     $s5,0($s0)
```

b) Assume that the memory addresses correspond to C variables (shown above) of type **int**. Now use your understanding of the above instruction sequence to translate the MIPS instructions into a corresponding C style assignment statement. Select the appropriate answer below.

Note: Enter the appropriate letter representing your selection.

a) Z = B*(Z+8) – (A-C)/2;
b) Z = B*(Z+8) – 2*(A-C);
c) Z = B*D + 8 + (A-C)/2;
d) Z = B*D + 8 + 2*(A-C);

Assembler Directives

5) (8 pts.) Examine the following C program variable declarations and translate them to the appropriate directives.

Note: Write the number of your desired selection.

a) short int x = 6;
1. x: .word 6
2. x: .half 6
3. x: .space 6
4. x: .align 6

b) unsigned char msg[8] = {1,4,9,7,3,6,8,2};
1. msg: .unsigned 1,4,9,7,3,6,8,2
2. msg: .byte 0x14973682
3. msg: .byte 1,4,9,7,3,6,8,2
4. msg: .half 1,4,9,7,3,6,8,2

c) int data[100];
1. data: .word 100
2. data: .space 100
3. data: .space 200
4. data: .space 0x190

d) char str[] = “hello\n”
1. str: .ascii “hello\n”
2. str: .asciiz “hello\n”
3. str: .byte “hello\n”
4. str: .asciiz “hello”

HLL to Assembly Translation

6) (30 pts.) Translate the following C code statements to an equivalent assembly language implementation.

Note: For constants/numbers writes your values in **decimal**. For registers, use the descriptive mnemonic (i.e. $t0, $s1). For labels, just enter it as shown. For opcodes, write the full opcode (e.g. b__ => enter ‘bgt’).

```
short data[20];
...
for(int i=0;i < 20; i++){
  data[i] = data[i] + 5;
}

// let i be stored in $t0
```

```
data .space    ________
     .text
     ...
     la      $s0,data
     add     $t0,____,______
     li      $t4,___________
L0:  slt     $t1,____,$t4
L1:  b___    $t1,$zero,______
L2:  sll     $t2,$t0,____
L3:  add     $s1,____,$t2
L4:  lh      $t3,________
L5:  addi    $t3,____,5
L6:  sh      $t3,________
L7:  addi    $t0,____,___
L8:  b       ________
L9:  ...
```

```
int *x, *y;
...
if(*x >= 10 && *y < 5)
  code A
else if (*x < 10 || *y > 5)
  code B
else
  code C
...
```

```
x    .space  4
y    .space  4
     ...
     la      $s0,x
     la      $s1,y
     addi    $t0,_____,10
     addi    $t1,_____,5
     lw      $s2,0($s0)
     lw      $s3,0($s1)
     lw      $t2,______
     lw      $t3,______
     blt     ____,$t0,____
     b___    $t3,___,____
L1:  Code A instructions
L2:  b       ___________
L3:  b___    $t2,$t0,____
L4:  b___    $t3,$t1,____
L5:  Code B instructions
L6:  b       ___________
L7:  Code C instructions
L8:  ...
```