Saishma Ramesh Kumar Date: . . .

CSCI 113 - Lab 4

1) int array A, 10 elements $s1 — A[0] → base case, swap A[4] and A[9]

```
lw $t0, 4($s1)
lw $t1, 9($s1)
sw $t0, 4($s1)
sw $t1, 9($s1)
```

$s1 = A1B2C3D4
$s2 = 5A6B7C8D

2) sra $t1, $s1, 3 → Shift Arithmetic Right by 3

↳ $t1 = 1010 0001 1011 0010 1100 0011 1101 0100 >>3
↳ $t1 = 1111 0100 0011 0110 0101 1000 0111 1010 → F436 587A

slr $t2, $t1, 1
↳ $t2 = 1111 0100 0011 0110 0101 1000 0111 1010 >>1
↳ $t2 = 0111 1010 0001 1011 0010 1100 0011 1101 ⟶ 7A1B 2C3D

3) sw $s1, 4($zero) → A1B2C3D4 (Stored in Memory[4])
sw $s2, 8($zero) → 5A6B7C8D (Stored in Memory[8])

Address	Data
[4]	A1
[5]	B2
[6]	C3
[7]	D4
[8]	5A
[9]	6B
[10]	7C
[11]	8D

lw $s1, 6($zero) → Memory[6]
$s1 = C3D45A6B → Content of $s1 in Hex number
Number of memory access made: 4

Date: . .

④ a)

Code Segments	Instruction Type	Addressing Mode
add $t1, $s2, $s1	R-type	Register addressing
lw $t0, 4($t1)	I-type	Base/Displacement addressing
bne $t0, $s5, End	I-type	PC-Relative addressing
addi $s1, $s1, 2	I-type	Immediate addressing
subi $s1, $s1, 1	I-type	Immediate addressing
j Start	J-type	Pseudo-direct addressing

b) lw $t0, 4($t1) (35, 8, 9)

offset: 0000 0000 0000 0100

100011 01001 01000

Hexadecimal: 1000 1101 0010 1000 0000 0000 0000 0100

↳ 8D280004

c) bne $t0, $s5, End offset: 0000 0000 0000 0011

000101 01000 10101

Hexadecimal: 0001 0101 0001 0101 0000 0000 0000 0011

↳ 15150003

5) The addressing mode used in jump is Pseudo-direct addressing. This type of addressing allows you to be able to jump anywhere as long as it's within the same block. Essentially, the address is calculated by taking 4 upper bits of the PC which is then concatenated to the 26 bit immediate value, and the lower 2 bits are 00, that is how the updating process of PC goes.