

Problem 1

Assembly Language Programming

In this problem, you will write a procedure that computes the maximum value of a 10 element array. The array begins at base address 200 and each element of the array is one word long. *Use only the registers described in the table below.* You should not require any registers in addition to those listed. Do not include assembler directives. Your answer should fit in the boxes provided. Use additional space only if necessary. Be sure to provide comments.

register	description	register	description
\$1	address of current element	\$4	value of current element of array
\$2	max value found so far	\$5	result (maximum value found)
\$3	branch predicate register	\$31	return address

Part 1.A To begin, write lines of code that initialize the element address register (\$1) to the address of the first element of the array and that initialize \$2 to the first element of the array.

label	instruction	comment
max:	addi \$1,\$0,200	# start with address 200
	lw \$2,0(\$1)	# \$2 gets first element

Part 1.B Write a code fragment that updates the element address register (\$1) to the address of the next element of the array. Then access the value of the current element and replace the maximum value found so far (\$2) if necessary.

label	instruction	comment
loop:	addi \$1,\$1,4	# update address
	lw \$4,0(\$1)	# get next element
	slt \$3,\$2,\$4	# is max < current elt?
	beq \$3,\$0,skip1	# if not, skip next line
	add \$2,\$0,\$4	# replace max w/ current

Part 1.C Finally, write a code fragment, such that if the last element of the array has not been processed, then loop back to continue looking for a maximum value. Otherwise, put the maximum value found into register \$5 and exit the procedure by returning to the caller.

label	instruction	comment
skip1:	slti \$3,\$1,236	# reached addr limit, yet?
	bne \$3,\$0,loop	# if not, loop
	add \$5,\$0,\$2	# put result in \$5
	jr \$31	# return to caller

Problem 2

Program Understanding

The following subroutine computes an interesting function.

address	label	instruction
1000	start:	addi \$2, \$0, 85
1004		addi \$3, \$0, 20
1008		jal mystery
1012	end:	j exit
1016	mystery:	add \$4, \$0, \$0
1020	loop:	andi \$5, \$2, 1
1024		beq \$5, \$0, skip2
1028		add \$4, \$4, \$3
1032	skip2:	srl \$2, \$2, 1
1036		sll \$3, \$3, 1
1040		bne \$2, \$0, loop
1044		jr \$31

Part 2.A (10 points) Consider the execution of the fragment of code beginning at `start:` and ending at `end:`. Assume all registers and data memory locations are cleared prior to the execution of the code sequence. List the results of the registers below *after the code fragment completes*.

\$2	<b>0</b>	\$3	<b>2560</b>	\$4	<b>1700</b>	\$5	<b>1</b>
-----	----------	-----	-------------	-----	-------------	-----	----------

Part 2.B (10 points) Determine the number of instructions executed for this code fragment (including the call of `mystery`). Count all instructions executed beginning at `start:` and ending at `end:`, inclusive. (show work)

**There will be 7 iterations of the loop, since there are 7 bits after the leading 0's in the binary representation of 85 in \$2, so it takes 7 shift rights to get \$2 to be 0. On each iteration, all instructions from 1020 to 1040, except 1028 are executed (7x5). The instruction at 1028 is executed on only 4 of the iterations, since there are 4 ones in the binary representation of 85. The 6 other instructions are executed exactly once.**

executed instructions:	$(7 \times 5) + 4 + 6 = 45$ instructions
------------------------	--

Part 2.C (5 points) Briefly describe the function computed by the program `mystery`.

**It's multiplying the values stored in \$2 and \$3, and placing the result in \$4.**

---

### Problem 3

### Assembly Language Programming

In this problem, you will write a procedure that computes the average magnitude of the elements of an 8-element array. (This is a function commonly used in image processing applications.) Your procedure will find the absolute value of each element (i.e., its magnitude), sum the magnitudes together, and then divide the result by 8. The array begins at base address 400 and each element of the array is one word long. *Use only the registers described in the table below.* You should not require any registers in addition to those listed. Do not include assembler directives. Your answer should fit in the boxes provided. Use additional space only if necessary. Provide comments.

register	description	register	description
\$1	address of current element	\$4	value of current element of array
\$2	running sum	\$5	result (average magnitude)
\$3	branch predicate register	\$31	return address

Part 3.A To begin, write lines of code that initialize the element address register (\$1) to the address of the first element of the array and that initialize \$2 to 0.

label	instruction	comment
avg-mag:	addi \$1, \$0, 400	# start with address 400
	addi \$2, \$0, 0	# initialize running sum

Part 3.B Next, take the current element, find its magnitude by computing its absolute value (i.e., if it is negative, negate it). Then add the magnitude to the running sum in \$2.

label	instruction	comment
loop:	lw \$4, 0(\$1)	# get next element
	slt \$3, \$4, \$0	# is it < 0?
	beq \$3, \$0, skip1	# if not, skip next inst.
	sub \$4, \$0, \$4	# otherwise, negate it
skip1:	add \$2, \$2, \$4	# add magnitude to sum

Part 3.C Finally, update the element address register (\$1) to the address of the next element of the array. If the last element of the array has not been processed, then loop back to continue summing elements. Otherwise, divide the sum by 8, put the result into register \$5, and exit the procedure by returning to the caller.

label	instruction	comment
	addi \$1, \$1, 4	# update address
	slti \$3, \$1, 432	# still below address
	bne \$3, \$0, loop	# if so, loop
	srl \$5, \$2, 3	# divide sum by 8
	jr \$31	# return to caller

Problem 4

Program Understanding

The following subroutine computes an interesting function.

address	label	instruction
1000	start:	addi \$2, \$0, 85
1004		addi \$3, \$0, 1
1008		jal mystery
1012	end:	j exit
1016	mystery:	add \$4, \$0, \$0
1020	loop:	andi \$5, \$2, 1
1024		beq \$5, \$0, skip
1028		add \$4, \$4, \$3
1032	skip:	srl \$2, \$2, 1
1036		bne \$2, \$0, loop
1040		jr \$31

Part 4.A Consider the execution of the fragment of code beginning at `start:` and ending at `end:`. Assume all registers and data memory locations are cleared prior to the execution of the code sequence. List the results of the registers below *after the code fragment completes*.

\$2	<b>0</b>	\$3	<b>1</b>	\$4	<b>4</b>	\$5	<b>1</b>
-----	----------	-----	----------	-----	----------	-----	----------

Part 4.B Determine the number of instructions executed for this code fragment (including the call of `mystery`). Count all instructions executed beginning at `start:` and ending at `end:`, inclusive. (show work)

**There will be 7 iterations of the loop, since there are 7 bits after the leading 0's in the binary representation of 85 in \$2, so it takes 7 shift rights to get \$2 to be 0. On each iteration, all instructions from 1020 to 1036, except 1028 are executed (7x4). The instruction at 1028 is executed on only 4 of the iterations, since there are 4 ones in the binary representation of 85. The 6 other instructions are executed exactly once.**

executed instructions:	$(7 \times 4) + 4 + 6 = 38$ instructions
------------------------	--

Part 4.C Briefly describe the function computed by the program `mystery`.

---

**It counts the number of 1's in the binary number in register \$2 and places the result in \$4.**

---