

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:		

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

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

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 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		\$3		\$4		\$5	
-----	--	-----	--	-----	--	-----	--

Part 2.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)

executed instructions	instructions
-----------------------	--------------

Part 2.C Briefly describe the function computed by this program.

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:		

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

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

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		\$3		\$4		\$5	
-----	--	-----	--	-----	--	-----	--

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)

executed instructions	instructions
-----------------------	--------------

Part 4.C Briefly describe the function computed by this program.
