ES215: Semester II [2021-2022]
Assignment 2
Due Date: 22-02-2022 (11:59pm) 100 points

Note: Make appropriate assumptions when required. State your assumptions, which do not contradict what has been given in the problem statement, clearly along with your solutions.
Late assignments will not be graded.
Please type up the solutions and submit them in PDF format. The source files have to be submitted separately. Please upload all the assignments in Google Classroom.
Please follow the honor code. Any violations will have severe repercussions.

---

1. Implement a function in MIPS32 assembly that computes the Highest value & index, Lowest value & index and Average value of an Array A[0..100]. Assume the base address of Array A is 200. Please state any other assumptions clearly. (**20 points**)

2. Consider a dual core processor. Program A completes in 6 seconds on core 1 and has a CPI of 6 and program B completes in 5 seconds on core 2 and has a CPI of 5. Both cores run at 1 GHz. What is the combined throughput of the processor? (**20 points**)

3. Say a processor X runs at 2 GHz and a program (Program A) executes 10 billion instructions with avg. CPI of 3. Compute how long this program ran for? Now, suppose we got another processor Y with a clock frequency of 4 GHz. Program A on this new processor executes 7 billion instructions with an average CPI of 5. What is the speedup (if any) of program A on processor Y over processor X? (**20 points**)

4. Consider a Processor with 1 GHz. Program A executes 9 billion instructions on this processor and has an average CPI of 1.5. With a newly designed processor that can run at 2 GHz, the execution time of Program A dropped to a quarter of the original value. What is the average CPI of Program A on the new design? (**20 points**)

5. Suppose a processor consumes 80 watts of power (Note: it is the total power) while running at 2 GHz and the operating voltage of this processor is 5V. (**20 points**)
    a) How much dynamic power would the same processor consume if the frequency were to be frequency to 5 GHz? (**10 points**)
    b) With technology scaling, consider that operating voltage drops to 2V. Now for the 2GHz processor, what is the overall power consumption with this design and what fraction of the power can be attributed to static power? (**10 points**)

Grace Question (overall 2 total points out of 10)

1. For any one of the programs implemented in Assignment#1, try to output the preprocessor, compiler and assembler generated codes on the traditional x86_64 and MIPS architectures.
    a. Suppose program name is “fib.c” then you need to share the corresponding preprocessed file (fib_x86.i, fib_mips.i), compiled code (fib_x86.s, fib_mips.s), assembled code (fib_x86.obj, fib_mips.obj) and binary codes (fib_x86.out, fib_mips.out)
    b. State your observations on the individual file sizes that you observe in each case and any specific observations that you can make.

Note your device would already have the x86/x86_64 toolchain, but you may need to additionally install the mips cross compiler toolchains.

MIPS Reference data card

Name	Register number	Usage	Preserved on call?
$zero	0	The constant value 0	n.a.
$v0–$v1	2–3	Values for results and expression evaluation	no
$a0–$a3	4–7	Arguments	no
$t0–$t7	8–15	Temporaries	no
$s0–$s7	16–23	Saved	yes
$t8–$t9	24–25	More temporaries	no
$gp	28	Global pointer	yes
$sp	29	Stack pointer	yes
$fp	30	Frame pointer	yes
$ra	31	Return address	yes

FIGURE 2.14 MIPS register conventions. Register 1, called $at, is reserved for the assembler (see Section 2.12), and registers 26–27, called $k0–$k1, are reserved for the operating system. This information is also found in Column 2 of the MIPS Reference Data Card at the front of this book.

MIPS operands

Name	Example	Comments
32 registers	$s0-$s7, $t0-$t9, $zero, $a0-$a3, $v0-$v1, $gp, $fp, $sp, $ra, $at	Fast locations for data. In MIPS, data must be in registers to perform arithmetic, register $zero always equals 0, and register $at is reserved by the assembler to handle large constants.
2^{30} memory words	Memory[0], Memory[4], . . . , Memory[4294967292]	Accessed only by data transfer instructions. MIPS uses byte addresses, so sequential word addresses differ by 4. Memory holds data structures, arrays, and spilled registers.

MIPS assembly language

Category	Instruction	Example	Meaning	Comments
Arithmetic	add	add $s1,$s2,$s3	$s1 = $s2 + $s3	Three register operands
	subtract	sub $s1,$s2,$s3	$s1 = $s2 – $s3	Three register operands
	add immediate	addi $s1,$s2,20	$s1 = $s2 + 20	Used to add constants
Data transfer	load word	lw $s1,20($s2)	$s1 = Memory[$s2 + 20]	Word from memory to register
	store word	sw $s1,20($s2)	Memory[$s2 + 20] = $s1	Word from register to memory
	load half	lh $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	load half unsigned	lhu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Halfword memory to register
	store half	sh $s1,20($s2)	Memory[$s2 + 20] = $s1	Halfword register to memory
	load byte	lb $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	load byte unsigned	lbu $s1,20($s2)	$s1 = Memory[$s2 + 20]	Byte from memory to register
	store byte	sb $s1,20($s2)	Memory[$s2 + 20] = $s1	Byte from register to memory
	load linked word	ll $s1,20($s2)	$s1 = Memory[$s2 + 20]	Load word as 1st half of atomic swap
	store condition. word	sc $s1,20($s2)	Memory[$s2+20]=$s1;$s1=0 or 1	Store word as 2nd half of atomic swap
	load upper immed.	lui $s1,20	$s1 = 20 * 2^{16}	Loads constant in upper 16 bits
Logical	and	and $s1,$s2,$s3	$s1 = $s2 & $s3	Three reg. operands; bit-by-bit AND
	or	or $s1,$s2,$s3	$s1 = $s2 | $s3	Three reg. operands; bit-by-bit OR
	nor	nor $s1,$s2,$s3	$s1 = ~ ($s2 | $s3)	Three reg. operands; bit-by-bit NOR
	and immediate	andi $s1,$s2,20	$s1 = $s2 & 20	Bit-by-bit AND reg with constant
	or immediate	ori $s1,$s2,20	$s1 = $s2 | 20	Bit-by-bit OR reg with constant
	shift left logical	sll $s1,$s2,10	$s1 = $s2 << 10	Shift left by constant
	shift right logical	srl $s1,$s2,10	$s1 = $s2 >> 10	Shift right by constant
Conditional branch	branch on equal	beq $s1,$s2,25	if ($s1 == $s2) go to PC + 4 + 100	Equal test; PC-relative branch
	branch on not equal	bne $s1,$s2,25	if ($s1!= $s2) go to PC + 4 + 100	Not equal test; PC-relative
	set on less than	slt $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than; for beq, bne
	set on less than unsigned	sltu $s1,$s2,$s3	if ($s2 < $s3) $s1 = 1; else $s1 = 0	Compare less than unsigned
	set less than immediate	slti $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant
	set less than immediate unsigned	sltiu $s1,$s2,20	if ($s2 < 20) $s1 = 1; else $s1 = 0	Compare less than constant unsigned
Unconditional jump	jump	j 2500	go to 10000	Jump to target address
	jump register	jr $ra	go to $ra	For switch, procedure return
	jump and link	jal 2500	$ra = PC + 4; go to 10000	For procedure call