# CS220 Mid-semester Examination

General instructions: Please write brief explanation for your answers. Answers without explanation will not receive any marks. If you submit multiple times, your last submission will be used for grading. Please provide an email address below where your responses can be sent.

Email address *

mainakc@cse.iitk.ac.in

Your name *

Mainak Chaudhuri

Your roll number *

0

Q1. Consider the following MIPS instruction stream where consecutive instructions are separated by semicolon: lb $2, 100($20); lw $1, 10($29); lhu $10, 90($12); add $3, $2, $2; addi $4, $1, -100; sub $5, $1, $10; add $11, $5, $2; sub $6, $4, $11. It is possible to regroup these instructions so that a group contains as many independent instructions as possible. As a result, the instructions in a group can be executed simultaneously in parallel preserving correctness. The groups are ordered based on the dependencies between two groups. A group of such instructions would first read the source register operands in parallel, then execute the operations in parallel, and finally, write the results to the registers. Once a group completes, the next group in the order can start. (a) Clearly write down the groups that can be formed from the given instruction stream and order the groups appropriately. You may name the groups in the order as G1, G2, etc.. Your answer must clearly specify which group contains which instructions. Each group must be as large as possible. Note that an instruction I2 is said to be independent of a previous instruction I1 if I2 does not need the result of I1. (8 points) (b) What is the minimum number of register file ports needed to execute these groups? How many of these ports are read ports, write ports, and read-write ports? (5+6 points)

(a) G1: lb $2, 100($20); lw $1, 10($29); lhu $10, 90($12)
G2: add $3, $2, $2; addi $4, $1, -100; sub $5, $1, $10;
G3: add $11, $5, $2
G4: sub $6, $4, $11

(b) Since reads from register file and writes to register file do not happen simultaneously, these ports can be shared between reads and writes. So, these shared ports are read-write ports. G1 needs three read-write ports. G2 needs two read ports and three read-write ports. G3 needs one read port and one read-write port. G4 needs one read port and one read-write port. Therefore, to execute these groups, it is sufficient to have two read ports and three read-write ports.

Q2. Consider designing an 4M x 8 bits SRAM module using 64 chips each being 32K x 16 bits. The environment views each 8-bit row of the SRAM module as a word. The environment inputs an address for a word to the module and the module outputs the corresponding 8-bit word. Internally, the words are laid out chip by chip meaning that the first chip is exhausted completely, then the second chip is used, and so on. Within a chip, the words are laid out row-wise. For example, within chip#0, the first row has word#0 and word#1, the second row has word#2 and word#3, the third row has word#4 and word#5, etc.. (a) How many bits of address does the environment send to the module? (2 points) (b) Given an input address, clearly specify which bits are used by the internal implementation of the module as the row address, column address, and chip select. Assume that in an n-bit address, bit#0 is least significant and bit#n-1 is most significant. Use this notation to specify your answer. (9 points)

(a) The environment sends a 22-bit address since there are 4M rows in the SRAM module. (b) The requested word resides completely in one chip. Since each chip contains a chunk of contiguous 64K words, the chip id where the requested word belongs to is address/64K i.e., the upper 6 bits of the address. Within a chip, a pair of words resides in a row. Within a row, the even column (i.e., column 0) has the even word and the odd column (i.e., column 1) has the odd word. So, the least significant bit of the address is the column address. The remaining middle 15 bits choose the row within a chip.
row address: bit#1 to bit#15
column address: bit#0
chip select: bit#16 to bit#21 (needs a 6 to 64 decoder to generate the actual chip selects)

Q3. Consider a 32-bit MIPS processor with an SRAM cache and a DRAM module. The DRAM contents are as follows. Each byte in the first 2^{22} addresses (i.e., 4M addresses) has value AB (in hexadecimal), each byte in the next 2^{22} addresses has value CD, each byte in the next 2^{22} addresses has value EF, each byte in the next 2^{22} addresses has value BA, each byte in the next 2^{22} addresses has value DC. Consider the MIPS instruction lh $2, -8($29), where the displacement is written in decimal.The value stored in $29 is 400000 in hexadecimal. (a) What value will $2 contain after this instruction completes execution? Express your answer in hexadecimal. Briefly explain. (8 points) (b) Suppose that the address encoded in the lh instruction is not found in the SRAM cache and therefore, a read request must be sent to the DRAM for fetching the data.The data interface between the SRAM cache and the DRAM controller is 64-byte wide. Assume that the DRAM controller always sends an address that is aligned to a 64-byte boundary to the DRAM module~(i.e., the address is divisible by 64). The DRAM module has two channels, eight ranks per channel, eight x16 chips per rank, eight banks per chip, and each bank has 8192 rows and 1024 columns, where each column is 16-bit wide matching the output width of a chip. Write down the address in hexadecimal that the DRAM controller will send to the DRAM to complete the lh instruction. (2 points) (c) The DRAM controller uses the address bits to decode the row number, rank number, bank number, and column number in that order from the most significant to the least significant side of the address. What are the row number, rank number, bank number, column number, and channel number corresponding to the address in part (b)? (10 points)

(a) The target address = 2^{22} - 8. Therefore, the target half-word belongs to the first section of 4M addresses. Thus the half-word is 0xABAB. This is sign-extended and put in $2. So, $2 contains 0xFFFFABAB.
(b) The address is 2^{22} - 8 aligned to 64-byte boundary (i.e., lower six bits would be zero). Since 0x400000 - 0x8 = 0x3FFFF8, the required address is 0x3FFFC0.
(c) Address = row, rank, bank, column-high, channel, column-low. We need 13 bits for row, 3 bits for rank, 3 bits for bank, 10 bits for column, 1 bit for channel. Now channel width is 128 bits or 16 bytes, since we have eight x16 chips in a rank. So, the lower four bits of the address is channel offset and the next two bits are column-low. Column-high is eight bits. Therefore, starting from the least significant side we have the following.
channel = 1
column = 1111111100 = 1020
bank = 111 = 7
rank = 111 = 7
row = 0000000000001 = 1

Q4. Consider a single-channel DIMM card. The channel has four ranks and each rank has 32 chips. The DIMM card uses x4 chips and each chip has eight banks. Each bank has 4096 rows. The capacity of each chip is 2 Gb. The DIMM card is connected to a DRAM controller which needs to respond with 16 bytes to the computer for each request. The computer does not have any SRAM cache and therefore, can only use the required portion of the 16-byte response and ignores the remaining portion. The DRAM controller uses the address bits to decode the row number, rank number, bank number, and column number in that order from the most significant to the least significant side of the address. An array A with N=2^{23} elements with each element being four bytes in size is stored in the aforementioned DIMM card starting at address zero. The array is accessed in a manner as shown in the following loop:[for (i=0; i<N; i++) { Access A[i]; }] What percentage of the accesses are row hits? Assume that a new request can be sent to the DIMM card only after the previous request has completed. Also, assume that the DRAM controller cannot reorder the sequence of requests coming from the computer. In other words, the requests are sent to the DIMM card exactly in the order of accesses shown in the loop. One request is sent per access shown in the loop. Initially, all the DRAM banks are in precharged state with no open row. (20 points)

Number of DRAM accesses = 2^{23}. The number of columns per chip = 2G bits/(8*4096*4 bits) = 16384. So, the size of a row per chip = 16384*4 bits = 2^{16} bits. On the first access, a row would be opened in each chip of a rank totaling to 2^{21} bits of open data. Since each access is 4 bytes or 32 bits, this open data can cater to 2^{16} consecutive accesses. Out of these, the first one would be a row miss. After these 2^{16} accesses, the bank number will change. Another 2^{16} such accesses would get satisfied by the new open row, out of which the first one would be a row miss. Thus, in each 2^{16} consecutive chunk of accesses, except the first one, all will be row hits. Thus, the total number of row hits = 2^{23} - 2^{23}/2^{16} = 2^{23} - 128.

Q5. Suppose we would like to represent the decimal real number -7.7 (note the minus sign) in IEEE 754 single-precision format. Derive the representation for each of the following four rounding modes: round toward positive infinity, round toward negative infinity, round toward zero, and round to the nearest with the IEEE 754 rule for halfway rounding. (12 points)

-7.7 = -111.1011001100110... = -1.111011001100110... x 2^2
Round toward positive infinity: -7.7 = -1.11101100110011001100110 x 2^2. Sign = 1, mantissa = 11101100110011001100110, exponent = 127+2 = 129 = 10000001 in binary.
Round toward negative infinity: -7.7 = -1.11101100110011001100111 x 2^2. Sign =1, mantissa = 11101100110011001100111, exponent = 127+2 = 129 = 10000001 in binary.
Round toward zero: same as round toward positive infinity.
Round to the nearest: same as round toward positive infinity.

Q6. Consider the following 32-bit MIPS instruction sequence with consecutive instructions separated by semi-colon. Assume that initially $1 contains 0x10000010 (in hexadecimal) and $2 contains 0x10000020. Initially, the word stored at address 0x10000010 is 0x12feabcd and the word stored at address 0x10000020 is 0x1abcdef2. What is the final hexadecimal value of the word stored at address 0x10000020? Assume 2's complement arithmetic throughout. (8 points) Instruction stream: [lb $1, 3($1); lhu $3, 2($1); sub $1, $1, $3; sh $1, 0($2)]

Corrected instruction stream: [lb $4, 3($1); lhu $3, 2($1); sub $1, $4, $3; sh $1, 0($2)]
lb $4, 3($1): $4 <---- 0xffffffcd (MIPS is big-endian; so byte at addr. 0x10000013 is 0xcd.)
lhu $3, 2($1): $3 <---- 0x0000abcd (half-word at address 0x10000012 is 0xabcd.)
sub $1, $4, $3: $1 <---- 0xffff5400 (result of subtracting $3 from $4)
sh $1, 0($2): 0x54 and 0x00 are stored at addresses 0x10000020 and 0x10000021
So, the word stored at 0x10000020 is 0x5400def2.