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 *

sohamg@iitk.ac.in

Your name *

Soham Ghosal

86/90

Your roll number *

180771

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)

Yes it is possible to group the instructions.

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

This ensures that all instructions in a group are independent of each other.

G1 needs 3 read ports and 3 write ports.(load requires 1 write and 1 read port)

G2 needs 5 read and 3 write ports(add and sub requires 2 read and 1 write port, whereas addi requires 1 read 1 write)
G3 needs 2 read ports and 1 write port
G4 requires 2 read ports and 1 write port

Minimum number of register file ports required: 5

Since each group is being done after the previous group has finished, we need a maximum of 5 read ports and 3 write ports for all the groups. But we can also use RW port, to reduce the total number of ports.

Therefore we finally require--> 8+5+6
3 read write ports and 2 read ports.

Initially while reading the source operands for G2, we use all 5 ports as read ports, and then we use the 3 RW ports for writing. This also works for G1 G3 and G4.

This works because the source operands are first read in parallel and only then are the write operations executed.

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)

We are implementing the SRAM using 32k*16 SRAM chip. Each chip has 2 chunks of 8 bits data, one on the left, one on the right.
We can arrange this in 64 ranks, each rank containing 1 chip.

2+9

In such a scenario, we will need log(32K)=15 bits for the row address, 1 bit for choosing between the left and the right subhalves in a chip(column address), and log(64)=6 bits for the rank select.
In total, we need 22 bits. Thus the environment sends 22 bits to the module.

Since, word#0 is in row#0 left half, and word#1 is in row#0 right half, we notice that the half changes most frequently, hence the column address must be the LSB.
Also, all the rows of a chip are exhausted before we move on to the next chip, ie. the next rank. Hence the next 15 bits from the lower significant side will be for the row address.
The most significant 6 bits will be for rank select.

Thus, bit#0 is for the column address(half select), bit#1 to bit#15 is for the row address, and bit#16 to bit#21 is for the chip select(rank select).

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)

400000 in hexadecimal = 2^22 in decimal
The half word addresses are therefore, 2^22-8 and 2^22-7
Both of these addresses contain 0xAB.

Therefore the half word will contain 0xABAB. Since the instruction is lh, this will be sign extended to 32 bits.
Therefore the final value in $2 will be 0xFFFFABAB

Data interface between SRAM cache and DRAM controller is 64 bytes wide.
Since we have 8 x16 chips per rank, channel width is thus 16*8=128 bits.
Thus, we require a burst length required is 4.

Now 2^22-8 is 0x3FFFF8
This address must be aligned to 64 byte boundary.
Thus the address sent to the DRAM module is
the hexadecimal encoding of 4194240, which is 0x3FFFC0

Binary corresponding to the sent address: 11111111111111111000000

The least significant 4 bits are for the burst length. The next 2 bits are column low.
The next bit is for the channel.The next 8bits are for column high(since total number of bits for column is log(1024)). Bank-->3 bits. Rank-->3 bits, row-->13 bits

Channel number=1(zero-based indexing)
Column number=1020
Bank number=7
Rank number=7
Row number=1

8+2+10

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)

2^31=rows*column bits*banks => coilumn bits=2^16
Thus number of columns=2^16/4 = 2^14
Channel width=128 bits=16 bytes

20

Least sig 4 bits of address has to be 0. The next 14 bits would be column number.The next 3 bits would be bank number. The next 2 bits would be rank number. The most sig 12 bits would be row number.

2^23 array elements.
Array size is 2^25 bytes.
This would use up the lower 25 bits of the address.Thus, only 2 bits from the row address is being used.Thus 4 rows of every rank,bank pair is being used.

The access pattern is: complete row 0 of rank 0 and bank 0, complete row 0 of rank 0 and bank 1...complete row 0 of rank 0 and bank 7...complete row 0 for rank 1 and bank 0...
Except first access to a row(row miss), all accesses would be row hits.{Since all banks are precharged}
There are 2^23 accesses, out of which 4*32 would be row misses and row conflicts.
(4 rows for each rank,bank pair and there are 32 such pairs)
Percentage of row hits = 100*(1-1/(2^16))

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= -111.101100110.... in binary.
In normalized notation, this is -1.11101100110...*2^2
Note that 0110 is recurring here.
Actual exponent=2, biased exponent=127+2=129
Sign bit=1
Since this number is negative, round towards zero and positive infinity will give the same result.

10

To round a negative number towards pos inf or zero, its magnitude must not be larger than what it is now.
positive infinity:(1)(10000001)(11101100110011001100110)
zero:(1)(10000001)(11101100110011001100110)
negative infinity:(1)(10000001)(11101100110011001100111)
Halfway rounding:(1)(10000001)(11101100110011001100110) [Since 23rd bit is 0,which is even]

This is not half-way rounding

Correct answer, but wrong reason

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)]

Note: question had been updated in chat

lb $4,3($1)--> $4 contains 0xFFFFFFCD(sign extended)
lhu $3,2($1)-->$3 contains 0xABCD(zero extended)
sub $1,$4,$3--> $1 contains 0xFFFF5400(two's complement subtraction)

6

The sh instruction will store the least significant 2 bytes of $1 in 0($2), ie. at the reqd address.

Final Hexadecimal Value at address 0x10000020: 0x5400 What is the word?