Vector Architectures

Bonus Assignment in IS2202 Computer Systems Architecture, 2022

Instructions

• The assignment must be solved *individually*.

• Solutions should be properly motivated, with a few short sentences and/or formulas.

• Answer the question in your own words. Copying complete answers from other sources is not acceptable. You may, however, quote other sources as part of an explanation or discussion. In that case, include a proper reference to the source and describe your interpretation.
Specifically, the description of your interpretation must be longer than the quote.

• For any material that is not your own: if you quote it or use it in any way, you must include a proper reference. This includes books, research papers, Internet sources, other students, and other material that is not your own.

• A proper reference describes the source clearly and concisely, so that any reader instantly can find the relevant page of the source. Author, title, date, and URL must be present, if available.

• All submitted reports will be automatically checked for plagiarism.

• Solutions may be given in English or Swedish, but not a mix of the languages.

• Only hand in PDF files from computer generated sources. *All text must be machine-readable.* Scanned text is not allowed.

• Solutions must be handed in through Canvas. Submissions by e-mail are not accepted.

• **You need to score at least 36 points (out of 46) to qualify for a bonus point at the exam**.

• The deadline is given in Canvas. After the deadline, submission is closed and no further submissions are possible.

Parameters for problems

020717

• You have a Swedish civic registration number (personnummer) of the form YyMmDd-XXXX.

Use the following formulas to calculate the parameters in the problems on the following pages.

• FP ADD and FP MUL Execution pipeline length PL =5 + d = 5+1=6 (stages)

• Number of Memory banks #B = $2^{(3+M)}$ = 2^(3)=8

• Bank busy time Tbusy = 4+D = 4+1=5 (ccs)

• Memory Bus access latency Taccess = 2+y = 2+2=4 (ccs)

Problem 1. Matrix Multiplication part I - 29 points

The KTH-student Malvina is implementing a Matrix multiplication on a vector machine with vector length 64, a single floating point MUL and a single floating point ADD-unit. The FP ADD and FP MUL both have execution pipelines that are *PL* stages long. The machine has a single load/store unit, capable of reading or writing one double-precision floating-point value at a time. The memory subsystem has *NB* banks, with a bank busy time of *Tbusy* clock cycles, and a memory bus access latency of *Taccess* clock cycles (i.e., the total access time for reading one double-precision FP value from memory is *Tbusy*+*Taccess* clock cycles).

```
/* variable declarations */
double A[64][64] ;
double B[64][64];
double C[64][64];
/* Matrix Multiply */
for (i = 0; i < 64; i=i+1)
      for (j = 0; j < 64; j=j+1) {
            A[i][j] = 0.0;
            for (k = 0; k < 64; k=k+1)
                  A[i][j] = A[i][j] + B[i][k] * C[k][j];
      }
```

a) Calculate how long it will take to complete 64-element vector loads with strides of 1, 2, 4, 8, 16, 32, and 64 on the given memory subsystem? (7p)
b) What is the stride when accessing A, B, and C? (3p)
c) Write the assembly code using VMIPS-style (or similar) instructions. (9p)
d) How many convoys are needed? How long is the execution time in ccs? (3p)
e) How long is the total execution time including overheads? (3p)
f) Consider the C-code for the multiplication above. Malvina decides to be smart and adds an element to the matrix C so that it is of the size [64][65]. What will the stride be for accessing C in this case, and how long will the execution time be for the algorithm including overheads (in ccs)? (4p)

Problem 2. Matrix Multiplication part II - 17 points

The KTH-student Osquar also implements the matrix multiplication on the same vector machine (vector length 64, etc). He decides to rewrite the matrix multiplication and use the mTvr-version of the matrix multiplication algorithm instead:

```
/* variable declarations */
double A[64][64] ;
double B[64][64];
double C[64][64];
/* Matrix Multiply */
for(j=0; j<64; j++) {
      for(i=0; i<64; i++) A[i][j]=0.0;
      for(k=0; k<64; k++)
            for(i=0; i<64; i++)
                  A[i][j] = A[i][j] + B[i][k] * C[k][j];
}
```

a) What is the stride when accessing A, B, and C? (3p)
b) Write the assembly code using VMIPS-style (or similar) instructions. (9p)
c) How many convoys are needed? How long is the execution time in ccs? (2p)
d) What is the total execution time in clock cycles, including overhead? (3p)
e) Compare the performance of Osquars and Malvinas implementations. Which implementation style was best? (1p)