CPSC 418	Advanced Computer Architecture	Spring 1995

Homework 5 Solutions

Note: I had the execution tables most of the way drawn, when the program froze on me and I had to kill it. So, in an effort to get the solutions out as soon as possible, I've sketched out the answers based on the information I calculated.

Problem 1 (0 Points)

Assembly code.

```
/* A, B : array [0 to 1023] of int (ints are 32 bits) */
for i = 0 to 1023 do
 A[i] = 3*A[i] + B[i]/A[i];
```

1(a) (0 Points)

Speed Demon:

```
        ; R1 = i       loop count
        ; R2 = (A)     base address of A
        ; R3 = (B)     base address of B
        ; R4 = (A[i])  address of i'th element of A
        ; R5, R6, R7   temporary values
        ;
500     ANDI  R1, 4096        ; R1 = i
504     ANDI  R2, 0x2000      ; R2 = (A)
508     ANDI  R3, 0x3000      ; R3 = (B)
50C     ADD   R4, R2, R1      ; R4 = (A[i])
510     ADD   R6, R3, R1      ; R6 = (B[i])
514     LDW   R5, R4          ; R5 = A[i]
518     LDW   R6, R6          ; R6 = B[i]
51C     DIV   R6, R6, R5      ; R6 = B[i]/A[i]
520     MULI  R5, R5, 3       ; R5 = A[i]*3
524     ADD   R5, R5, R4      ; R5 = A[i]*3 + B[i]/A[i]
528     STW   R4, R5          ; A[i] = A[i]*3 + B[i]/A[i]
52C     SUBI  R1, R1, 4       ; i = i - 4
530     BGE   R1, -20
```

1(b) (0 Points)

Brainiac:

```
        ; RBR = i      loop count
```

```
        ; R2 = (A)      base address of A
        ; R3 = (B)      base address of B
        ; R5, R6        temporary values
        ; R1, R4, R7    not used
500     LDBR  1023              ; RBR = i
504     ANDI  R2, 0x2000        ; R2 = (A)
508     ANDI  R3, 0x3000        ; R3 = (B)
50C     LDW   R5, (R2)[RBR]     ; R5 = A[i]
514     LDW   R6, (R3)[RBR]     ; R6 = B[i]
518     DIV   R6, R6, R5        ; R6 = B[i]/A[i]
51C     MULI  R5, R5, 3         ; R5 = A[i]*3
520     ADD   R5, R5, R6        ; R5 = A[i]*3 + B[i]/A[i]
524     STW   (R2)[RBR], R5     ; A[i] = A[i]*3 + B[i]/A[i]
528     BCT   -18
```

Problem 2 (0 Points)

Instruction counts:

	static	dynamic
code speed demon	13	$3 + (1024 * 10) = 10243$
code brainiac	10	$3 + (1024 * 7) = 7171$
memory speed demon	$13 + 3$	$3 + 1024(10 + 3) = 13315$
memory brainiac	$10 + 3$	$3 + 1024(7 + 3) = 10243$

Problem 3 (0 Points)

Execution tables

Things to look for:

- Loads spend a cycle in the LS stage, and then go to Writeback, this causes a normal arithmetic/logical instruction immediately following a Load to stall for one cycle in EX waiting for WB to become available.
- Similar to the structural hazards caused by loads, divides spend two cycles in Divide and then transfer to WriteBack.
- Remember that the Divide stage does not have a bypass path.
- When the FI stage stalls, the PC cannot increment.

Problem 4 (0 Points)

CPI : CPI = total cycles / total instruction count

Total cycles can be determined from the execution tables and knowing that the loop is executed 1024 times.

The total instruction count was determined in problem 2.

Doing the calculations for the speed demon:

If in the execution table, one execution of the loop takes 13 cycles, then the total cycles is: $3 + 1024 * 13 = 13315$, which gives a CPI of: $13315/10243 = 1.30$.

Problem 5 (0 Points) One out of every sixteen references to B will cause a cache miss (each cache miss brings in 128 bytes, or 16 words, from the L2 cache to the L1 cache). Thus, 64 out of the 1024 references to B will spend five cycles (rather than 1) in LS. Depending on how your code is written, this will most likely cause the subsequent instructions to stall for four additional cycles. Thus, the total cycles for 64 of the 1024 iterations will increase by four cycles.

Note: In lecture I mistakenly said that one out of every four iterations will miss in the L1 cache. The correct number is one out of every sixteen.

Total cycles $= 3 + 960 * 13 + 64 * (13 + 4) = 13571$

CPI = 13571 / 10243 = 1.32

Problem 6 (0 Points)

Time = TotalCycles * ClockCycle

For the example we have been following:

13571 * 1/(200E+6) = 67.9 microseconds

Then use Speedup = (TimeSlow - TimeFast) / TimeSlow

Problem 7 (0 Points) In general, this was a realistic example, except that divides should take longer. Having divides take longer would just make the execution tables more tedious to draw. Also, for those who used displacement addressing mode because the address of A and B was small enough to fit, in general the address of data will not fit into a sixteen bit displacement field.

The observant instruction set architect will notice that if the Brainiac uses fixed width instructions, the indexed addressing mode will require a very small (few bits) displacement field, because there are already three registers that must fit into the instruction word.

One note: when compiling a program we can't know where in the memory hierarchy (L1 cache, L2 cache, main memory, etc) data is stored, so trying to improve the performance of a program by writing code with the assumption that we know where data is will not work.

Problem 8 (0 Points)

Both the indexing mode and BCT instruction turned out to be useful in the Brainiac. But, as was shown in HW-3 prob 2, a complex addressing mode is not always the best solution. Both features reduced the number of instructions and registers required.

Problem 9 (0 Points)

Index addressing mode

- The EX stage needs to perform a shift by 1 (half-word indexing), 2 (word indexing), or 4 (double-word indexing), an addition, and a multiplication. In two cycles it could do two of the three operations w/o any cost. To perform all three operations though, we will need increased hardware in the EX stage and/or an increased cycle time.

- As mentioned earlier, the displacement (immediate) field will have to be fairly small in order to fit the op-code, three registers, and the displacement field in a single instruction word.

BCT instruction/RBR reg

- We will need additional hardware for the register RBR.
- RBR requires a special decrementer (addtional hardware), because it does not use the EX stage.
- The DI instruction will have to do more work because the BCT instruction requires specialized functionality. This will require additional hardware and possibly increase the cycle time.
- RBR can be an input to EX, this will require additional signals (wires) to EX and an additional multiplexer. These may increase the cycle time.

Problem 10 (0 Points)

This is not a good benchmark, because:

- The program is too short, doesn't show affects of I-cache, register allocation, etc. Even the data — just two 1k arrays is pretty small.
- Instruction in arithmetic functions (one divide, one multiply, several adds) mix doesn't reflect real programs. Also, this is purely integer, should also test floating-point performance.