**Shishir Ghorashainee**

CSC 362 Homework #5

Due Date: Wednesday, October 23

Word process all answers. You must show your work for questions 4, 6, 7 and show your work for other problems for partial credit.

1. Answer the following questions regarding the Intel x86 and PowerPC instruction sets (refer to the figures on slide 3 of the chapter 5 power point notes).
   1. How many bits can be used to specify the op code in an instruction for the x86 processor?

**1 byte= 8bits or 2 bytes=16bits**

(this is the largest number available)?  **(2^16)**

How many op codes could this support? **1 byte= 2^8 or 2 bytes= 2^16**

* 1. How many registers does the PowerPC have? **(5bits per register), so it’s 32 Registers.**

You can determine this by looking at the number of bits reserved for register numbers.

How many registers are used in load/store instructions for the PowerPC? **5** There is a range, what is the least and what is the most number? **(2 is the least and 3 is the most number)**

* 1. What is the largest immediate datum that can be specified for an instruction in the x86 processor? **(2^31) - 1**

For the PowerPC? **(2^15) - 1**

1. We decide to add an immediate addressing mode to MARIE, as in Load #1 and Add #10. Assume the immediate datum will be a two’s complement integer. Answer the following questions.
   1. What are the range of values that we can specify using this mode?

**Range:** -2^11 …… 2^11-1

= -2048 ……. 2047

* 1. We add two instructions to MARIE, LoadImm and AddImm. This gives us 17 instructions. Why is this an issue?

**When we’re given 17 instructions, the operand will change from 4 bits to 5 bits. And, if we’re going to increase the instruction to 17, then the memory is going to be 4k x 17, which would increase IR, Data bus, AC, MBR and ALU. If we want to keep it at 16, we’d need to change the instruction format which will reduce our MAR, PC, Address bus.**

* 1. We could remove one instruction from MARIE because of these two new instructions, which instruction and why?

**= The instruction to be removed will be clear because the LoadImm would just load the value to the AC. For instance, LoadImm #0, this would put AC to 0, which is what clear does. So clear becomes not useful in this case.**

1. We decide to expand MARIE from a 4Kx16 memory to a 32KxY memory (you will have to figure out Y yourself). The enlarged memory means the operand size has to increase, which will increase the length of the instruction. Answer the following.
   1. What is the new instruction format?

**Instruction format: 4 - 15**

What is the new value for the word size (Y)? **New value for the word size (Y) is 19 bits.**

* 1. Hardware will have to change. For each of these indicate its new size or that it does not need to change size.

i. PC **= 15 bits** ii. MBR **= 19 Bits** iii. MAR **= 15 Bits** iv. AC **= 19 Bits**

v. ALU width **= 19 Bits** vi. InReg & OutReg =**Unchanged** vii. Address bus **= 15 Bits**

viii. Data bus **= 19 Bits** ix. Status flags register **= Unchanged**

1. Assume a CPU with a fixed 24-bit instruction length has the following instruction format: opcode [num operands] [mode1 operand1] [mode2 operand2]

The num operands field specifies whether there are 0, 1 or 2 operands. Each operand has its own mode specifier. Assume the CPU’s instruction set consists of 100 instructions and 6 addressing modes. Answer the following questions.

Given,

24-bit instruction.

CPU instruction set consists of 100 instructions and 6 addressing modes.

They sum up to 24 bits.

Op code = Log2(100) = 7 bits (approx.)

Number of operands = 2 bits

Mode = Log2(6) = 3 bits (approx.)

* 1. If an instruction uses two registers, how many registers can be referenced?

**24-7-2-3-3 = 9 bits**

**We have two registers so,**

**9/2 = 4.5 bits per register, but we cannot use 0.5 of a bit. So, it’s 4 bits for each register specifier.**

**Therefore, it is going to be 2^4 = 16 registers**

* 1. If an instruction uses one register and one immediate datum (in two’s complement), what is the largest immediate datum that can be referenced assuming there are 8 registers?

= 24 - 7 - 2 - 3 - 3

= **9 bits** for the register & immediate datum.

We have 8 registers, so the register takes 3 bits, leaving 6

bits for the immediate datum.

What’s the largest immediate datum in 6 bits? **= 31**

**-2^5 . . . . . + 2^5 -1**

* 1. If an instruction has one operand, an address which consists of an index register and a displacement (an unsigned binary number), what is the largest displacement available if there are 4 index registers?

**24 - 7 - 2 – 3 = 12 bits for the operand, memory addresses**

**are stored in unsigned integers, largest unsigned integer in 24-12-2=10 bits**

**is 2^10 - 1 = 1023**

* 1. If an instruction has one operand, a direct memory address (an unsigned binary number), what is the largest address that can be specified?

= **24 - 7 - 2 - 3 = 12 bits.** With 4 registers, 3 bits for

for the register, 12 bits for the address. **Largest address?**

**2^12 - 1 = 4095**

1. Assume the following values are in memory as shown below to the right, and register R1 is the index register and is storing 10000. What is loaded into the accumulator with the instruction “Load 12000” given each of the following addressing modes?

Address Datum

12000 20000

* 1. Immediate:**12000** … …
  2. Direct : **20000** 15000 12000
  3. Indirect : **25000** … …
  4. Indexed : **15000** 20000 25000

… …

22000 15000

… …

25000 10000

1. We are comparing two processors’ performances where both have an 12-stage fetch-execute cycle, one processor is pipelined and the other is not. Assume *tp* = 1.5 and we run a program of 100,000 instructions on both processors. To compute how much faster the pipelined machine is, use cycles of slower machine / cycles of faster machine.
   1. How much faster is the pipelined processor assuming no stalls arise?

=**For non-pipelined and for pipelined processor (100000\*12)/((100000+12-1)\*1.5) = 1200000/150016.5 = 7.9991. Therefore, Pipelined processor assuming no stalls is 7.9991 times faster.**

* 1. How much faster is the pipelined processor assuming there is 1 stall every 4 instructions?

There are 1 stall every 2.5 instructions means 0.4 per instruction and now calculation 500000/84011= 5.95 (approx.6). So, it is 5.95 times faster.

**Stalls = 100000/4 = 25000**

= **There is 1 stall every 4 instructions which means 0.25 per instruction and So,**

**= (100000\*12)/((100000+12-1+25000)\*1.5)**

**= 1200000/187516.5**

**= 6.3994 (approximately 6). So, it is 6.399 times faster.**

* 1. How much faster is the pipelined processor assuming there are 2.1 stalls every instruction?

= **There are 2.1 stalls every instruction of pipelined processor 2.1\* 100000 = 210000. So, =(100000\*12)/((100000+12-1+210000)\*1.5)**

**= 1200000/465016.5**

**= 2.58055.**

**Therefore, it is 2.58 times faster.**

* 1. Bonus question: How many stalls *per instruction* would be required so that the two processors had equal performance?

**=Let Stalls be S.**

**Then, for two processors with equal performance; 1200000 = (S + 100011) \*1.5**

**S = 699989**

**699989/100000 = 6.99989 stalls per instruction would be required.**

1. Two processors have 7-stage fetch-execute cycles where branches are determined in **stage 4**. One processor is not pipelined, and the other is pipelined. Assuming that *tp* = 1, answer the following questions when running a program with 25,000 instructions where 2,000 of the instructions are conditional branches and each branch, if taken, skips over 5 instructions.
   1. How much faster is the pipelined machine over the non-pipelined machine assuming that no branches are taken.

**=If no branches are taken, then the pipelined machine takes n + k – 1 cycle or 25,000 + 7 – 1 = 25,006 cycles. The non-pipelined machine takes n \* k cycles or 25,000 \* 7 = 175,000 cycles. The pipelined machine is 175,000 / 25,006 = 6.9983 times faster (almost 7 times faster).**

* 1. How much faster is the pipelined machine over the non-pipelined machine assuming that all branches are taken.

=**If all branches are taken, then n drops from 25,000 to 25,000 – 2,000 \* 5 = 15,000. So, our non-pipelined machine’s performance becomes n \* k = 15,000 \* 7 = 105,000 cycles. The pipelined machine has fewer instructions, but each branch accrues a branch penalty. Since branches are computed in stage 4, the branch penalty is 3 cycles. Our new formula is n + k – 1 + b \* 3 where b is the number of branches. So we have 15,000 + 7 – 1 + 2,000 \* 3 = 21,006. Our pipelined machine is still much faster by a factor of 105,000 / 21,006 = 4.99857 but the speeds are getting (slightly) closer.**

* 1. Bonus question: Instead of the program having 2,000 branches, how many branches would the program have to have assuming every branch is taken (and every branch skips over 5 instructions) for the non-pipelined machine to execute the program in the same time as the pipelined machine?

**(25000 –branches \* 5) \* 7 = (25000 –branches \* 5 + 7 – 1 + branches \* 3) \* 1**

**175000 – 25006 = 33 \* branches**

**branches = 149994 / 33**

**branches = 4545.272727 which rounds to 4545**