COMP4300
Computer Architecture, Fall 2017
Midterm Examination
October 5, 2017

NAME KEY

(If you are unsure of the answer to any question, tell all your reasoning for giving the answer you did, for partial credit)

1. (10 points) Suppose that a given machine spends 1/3 of its execution time AFTER OPTIMIZATION doing jump instructions. The design team for the manufacturer optimized the calculation of target addresses by jump instructions. If the optimization resulted in the speedup of processing jump instructions by a factor of 4, what fraction of the execution time did the machine spend on jump instructions BEFORE OPTIMIZATION?

$$(1 - T_j) = 2\left(\frac{T_j}{4}\right)$$

$$2 - 2T_j = T_j$$

$$2 = 3T_j$$

$$\boxed{T_j = 2/3}$$

(Makes sense – after speedup spends 1/6 original time on jumps, 1/3 on non-jumps)

2. PDP-8 assembly language
    a. (7 points) What is the two's complement 12-bit representation of negative 17?

    b. (13 points) The following PDP-8 machine language program loads the contents of address 0x400 into the accumulator, subtracts 17 from it, and stores it back to address 0x400. Use address 0x711 to store the address 0x400. Fill in the blanks in the binary. See last page for PDP-8 opcodes. Note that all numbers on the PDP-8 programming card are in *octal,* while numbers in the question below are either hexadecimal (if prefixed by 0x) or binary (if made up of zeroes and ones). Each blank is one binary digit.

Addr	**Mnemonic**	**Binary**
0x700	CLA	111 010 000 000
0x701	TAD I 0x711	001 1 10 010 001
0x702	TAD I 0x712	001 010 010 010
0x703	DCA I 0x711	011 110 010 001
0x704	HLT	111 100 000 010
0x711	(addr stored here)	010 000 000 000
0x712	(data stored here) -17	111 111 101 111

0x711 = 0111 0001 0001
0x712 = 0111 0001 0010

0x400 = 0100 0000 0000

3. (3 x 7 points)
   a. List two ways a RISC processor differs from a CISC processor?

      Load/store
      GPR's
      simple instr. format

   b. Give two reasons (and for each reason explain why it is a reason) RISC processors have largely replaced CISC in new CPU designs?

      better compilers
      bigger chips
      faster chips

   c. There is one big exception to the rule that "CISC has been replaced by RISC". What architecture is that? Why has it not been replaced by RISC?

      80x86
      Backwards compatibility

4. (8 points)
   a. In the following VHDL process, if input A changes at time 45nS and no other inputs change after that time , at what time will all the output signals be *guaranteed* to have assumed their final value, regardless o f their initial values?

```
foo: process(A,B,C) is
begin
  if A = '1' then
       X <= '1' after 35 ns;
  else
       Y <= '1' after 10 ns;
  end if;
  Z  < = '1' after 5 ns;
  W <= '0' after 30 ns;
end
```

$\boxed{80nS}$

b . (5 points) Where in a VHDL description of a piece of hardware would the code for the process above be used? In an entity declaration or an architecture declaration? Why?

archicktecture

entity only defines ports +
generic parameters

5. Consider the following computer system, which has a load-store architecture:

   -ALU operations take 1 cycle and make up 40% of the dynamic instruction count

   -Floating point instructions are 6 cycles, and make up 5%

   -Branches take 5 cycles and make up 25%

   -Loads and stores are 3 cycles and make up the remaining 30%

   a. (7 points) What is the average CPI?

$$\text{AVG CPI} = \sum_{i \in \text{Type}} CPI_i \cdot Freq_i = 0.4\times 1 + 0.05\times 6 + 0.25\times 5 + 0.3\times 3$$

$$= .4 + .3 + 1.25 + 0.9$$

$$= 2.85$$

   b. (10 points) Optimization A reduces cycle time to 0.8 of the original cycle time and speeds up branches by 1 cycle. Optimization B speeds up floating point operations by 50% but lengthens the original cycle time by 10%. Which optimization improves performance more?

$$\text{Time}_A = (0.8CT)\times(CPI_A)\times IC$$

$$CPI_A = 0.4 + 0.3 + 4\times 0.25 + 0.9 = 2.6$$

$$\text{Time}_A = 0.8\times 2.6\times CT\times IC = 1.6\times CT\times IC$$

$$\text{Time}_B = (1.1CT)\times(CPI_B)\times IC$$

$$CPI_B = 0.4 + 0.05\times 3 + 1.25 + 0.9 = 2.7$$

$$\text{Time}_B = 1.1\times 2.7\times CT\times IC = 2.97\times CT\times IC$$

$$\boxed{\text{A is better}}$$

6. (10 points) Of the following architectures we have discussed, which one is likely to have the most complicated instruction decode mechanism, and why? PDP-8, MIPS, or Z-80?

$\boxed{\text{Z-80}}$: Instructions are variable length, requiring more complex logic to decode + to calculate address of next instruction

7. (9 points) Suppose you are a computer buyer in the 1990's while Moore's law still accurately described the annual increase in processor speed (41% per year). You are interested in achieving a 2x speedup in executing a particular program, because your company will make more revenue if you can do it. A competitor to the maker of your current computer sells a machine right now that is 3x as expensive as your current machine, but has 2x the speed on this program (half the cycle time). Both machines execute the same ISA. Your foregone revenue each year, if you don't speed up the calculation, is the cost of re-buying your current machine (in this year's dollars -- ignore inflation). Which is cheaper at the end of two years? Foregoing the revenue until you can buy a 2x faster machine from your competitor, or buying the competitor's machine right now? Do not include the cost of buying your current machine in your calculation – it is a sunk cost and does not figure in your calculation. You can sell your current machine on the used market for half its purchase price right now if you replace it with the competitors faster machine.

x = cost of current machine

Buy new: $3x - 0.5x = 2.5x$

Forego: $2x \leftarrow \boxed{\text{cheaper to Forego}}$