463-1 Fall 2017   REVIEW SHEET FOR FINAL

As before, the exam will include conceptual questions and problems to work.

The conceptual questions below are a complete guide to the conceptual part of the exam. However, the format will probably be different. The exam may include any of the following: multiple choice, short answer, true/false and similar types of questions.

The homework problems are your best guide to the problem section of the exam.

Material that is not on the homework or the review sheet will not appear on the exam, i.e., there are a few things in the slides that are just background information for you.

As before, I will give you a chart of powers of two. If an Ascii chart is needed, that will be provided also.

As on the previous exam, simple calculators, preferably scientific ones, are permitted but not necessary. Cell phones and other electronic devices are not permitted. For detailed calculator rules, see the review sheet from the previous exam.

Conceptual questions will be based on the assigned reading, listed here for your convenience:

- 2jmnpqr
- 5abcd
- 6ad

Problems to work will be based on the following assignments:

- HW7
- HW8
- HW9

Make sure you understand the questions and aren't just following a pattern. For example, the questions involving overflow, pipelining, etc. may have a different format than on the homework.

Part I. ISAs

1. What is an instruction set?

2. What is an ISA? What are the some important features that can be used to characterize an ISA?

3. Explain how stack, accumulator, and register architectures work and how they differ. What are their advantages and disadvantages? Which one is most common today?

4. A stack machine and an accumulator machine both commonly have one address. How do they differ?

5. What is postfix (reverse Polish) notation? Which type of architecture requires it? Why?

6. What does GPR stand for? What are three types of GPR machines? How do they differ?

7. How do 2-address and 3-address register architectures differ?

8. How does the ISA affect the instruction length? Do longer instructions necessarily lead to longer programs? Why or why not?

9. What is a compute-bound program? I/O-bound program? Why is this distinction important?

10. What is the maximum number of fetch operand steps that instructions in each of these machines can have? Explain why for each case.

a) A stack machine.
b) An accumulator machine.
c) A memory-memory machine where instructions have at most 2 operands.
d) A memory-memory machine with some 3-operand instructions.
e) A register-memory machine where instructions have at most 2 operands.
f) A register-memory machine with some 3-operand instructions.
g) A load-store machine.

11. A stack machine can compute any arithmetic expression without using temporary variables. An accumulator machine or a register machine cannot. Explain why. Hint: how would each of these machines calculate a formula such as (a+b)/(c+d) ?

Part II. Byte ordering

1. What is the difference between little-endian and big-endian architecture? Give an example of each type of machine.

2. Why is endianness important in I/O and networking?

Part III. Pipelining

1. What is pipelining? Why does it speed up execution? Why can't we keep the pipeline full all the time?

2. What is a pipeline hazard? a stall? flushing the pipeline?

3. What are three kinds of pipeline hazards? Define them.

4. How can an optimizing compiler eliminate some types of pipeline hazards?

Part IV. Memory

1. What is RAM? What is ROM? What is each used for? How are they different? What does "volatile" mean? Which one(s) are volatile?

2. What is a PROM? EPROM? EEPROM? What is flash memory? What is each used for?

3. What is SRAM? What is DRAM? What is each used for? How are they different?

4. Where is the fastest memory usually located in a computer? Why?

Part V. Cache

1. What is the purpose of a cache? Describe the algorithm for memory retrieval when cache is present.

2. What is the principle of locality? Give two reasons that good quality software might still violate the principle of locality.

3. Why is cache stored and updated a block at a time instead of a byte at a time?

4. Define hit, miss, hit rate, miss rate, hit time, miss time, miss penalty. How can you calculate the miss rate from the hit rate? How do you calculate the miss penalty?

5. What is the EAT? How do you calculate it? Why does a small change in the hit rate make such a big change in the EAT?

6. What is a replacement policy? Why is it important?

7. What is a dirty block? What is a write policy? How does write through work? How does write back work? Give an advantage and disadvantage of each. Why is write back a problem in multiple CPU configurations? Why is write through still valuable even though it updates memory with each cache write?

8. What is a multilevel cache? Why is it used in almost all of today's computers? What is a typical number of levels of cache in a PC?

9. What is a strictly inclusive cache? What is an exclusive cache?

Part VI. Floating point

1. What are some advantages of floating point over fixed point?

2. Why is floating point not generally used in accounting?

3. How many bits are in each field in the IEEE 754 single precision standard?

4. What is the purpose of the bias? How is the bias calculated?

5. What is floating point normalization? What is the main benefit of normalization? What is the extra benefit we get from not representing the initial 1?

6. Make sure you can recognize the 5 special cases on slide 2j-71a. What is NaN? Why is it useful to have a specific value to represent it? Why is it useful to have a specific value to represent infinity?

7. What are denormalized numbers? How do denormalized numbers allow for a smooth transition from the smallest normalized number to 0? Why is that a good thing?

8. How can you recognize a denormalized number? What value does an exponent of 0 represent? Why don't denormalized numbers have an explicit 1? (Hint: what is the smallest regular exponent?)

9. Why is floating point error unavoidable?

10. Why is it not safe to compare floating point numbers using ‘==’? What is a workaround for this problem?

11. What is floating point overflow? What is floating point underflow?

12. What is the range of a representation? What is accuracy? What is precision? How does normalized floating point allow us to maintain both precision and accuracy to the extent possible, i.e., why would we lose both precision and accuracy if we just encoded floating point numbers directly from their representation in 2j:58? How does denormalized floating point allow us to maintain both precision and accuracy to the extent possible for very small numbers, i.e., what would happen if we didn’t have denormalized floating point as an option?

Part VII. Unicode

1. How many characters can be represented in ASCII? Why are there multiple “extended ASCII” schemes? What is wrong with having multiple extended ASCII schemes?

2. What is Unicode? How does Unicode solve the “extended ASCII” problem?

3. What is a Unicode code point? How many of them are there? How are they written?

4. What is a Unicode code plane? How many of them are there? How many code points are there in a code plane?

5. Which code plane is the basic multilingual plane? Which code points does it contain? Why are some code points omitted, i.e., why is the BMP discontinuous? Why is the BMP important, i.e., why do most Unicode users never need to look outside the BMP?

6. Which code points match the 7-bit ASCII characters? Which standard do the code points from 128 to 255 match?

7. What is the difference between Unicode and UTF? What is UTF-8? What is UTF-16? What is UTF-32? What is self-synchronization and why is it useful? (You won’t have to do conversions or computations based on the charts in the slides.)

8. Why are there multiple UTF options? Which one(s) are fixed length? Which one(s) are fixed length for code points in the BMP? Why are fixed length formats useful? Why are variable length formats useful?

9. Which UTF formats have an endianness issue? Why? What is a BOM (byte order mark)? Is a BOM always required? What is the default if a file has no BOM and there is no other way to identify the encoding?

10. Some software manufacturers provide a “BOM” on UTF-8 files. Why is this not truly a BOM? What problems does this cause?

11. What does Microsoft do with UTF-16 files that is different from the standard? What problems can this cause?

12. Which encoding is most prevalent on the Web? What format do each of the following use internally: Linux, Windows, Java?

13. There are various websites that contain information about each Unicode code point. For example, if you google “Unicode e with acute accent”, you will find its official name (LATIN SMALL LETTER E WITH ACUTE), its code point number, and its representation in UTF-8, UTF-16, and UTF-32. (Note: use a real Unicode website for this question, not something like Emojipedia.)

a) Is this symbol Ascii? Is it in the BMP? Do the UTF-8, UTF-16 and UTF-32 representations given for this character on the website have the correct number of characters as specified in the charts in section 2r?

b) Do the same thing for the character “white smiling face.” (Note: in older Unicode terminology, “white” means an outline character, “black” means a filled-in graphic, and “heavy” means a character with a thicker outline. These terms are no longer used for new characters, but they do not have anything to do with race.)

c) Do the same thing for the character “turkey”.

d) Which of these characters are available in ISO-8859-1?