Approximately 50 points of the test will cover the material that could have been tested in the first midterm. Those questions will have formats revealed in Practice Midterm #1 and Midterm #1. The remaining points will cover chapters 3, 4, and 5 as well as p3 and p4. Examples of questions from these chapters are:

1. (25 points, all or nothing) In the CPU.circ you designed for p4, how did your circuit deal with subtraction opcodes need to subtract? Describe your components, wiring, and reasoning in enough detail so that we can determine that you actually designed your circuit. This may only take two or three sentences. You may use a drawing to help with your explanation.

2. (10 points) What is the goal of the memory hierarchy? What principle makes it possible to achieve this goal? Give the two types, and explain what they are.

3. (10 points) What is Cache Coherence, and why does it matter? Is it a concern only in parallel computers? Why or why not?

4. (10 points) Caches can be either Virtually Addressed or Physically Addressed. Explain the difference, and give one advantage and one disadvantage to using Virtually addressed caches.

5. (51 points) Consider a memory system that uses a 24-bit address, is byte addressable, and each line in cache holds 16 bytes of data.
   a) (18 points) Assume a direct-mapped cache with a tag field in the address of 12 bits. Answer the following questions. Please state the address format in terms bit ranges, e.g “block: 0-5” where 0 is the LSB. Showing your work may earn you partial credit.
      1) (3 points) Number of lines in the cache:
      2) (3 points) Size of the cache (in data bytes):
      3) (3 points) Size of the tag in bits:
      4) (3 points) Address format:
      5) (6 points) We are trying to read a byte with address 0x49BE51.
          What line number(s) (in hex) would we look at?
          If the byte is the cache, what is the tag (in hex) of its line?
   b) (12 points) Assume a fully-associative cache.
      1) (3 points) How big is the tag in bits?
      2) (3 points) Address format:
      3) (6 points) We are trying to read a byte with address 0xAB1483.
          What line number(s) would we look at?
          If the byte is the cache, what is the tag (in hex) of its line?
   c) (21 points) Assume a 2-way set associative cache with 8K lines. Answer the following questions. Please state the address format in terms bit ranges. Showing your work may earn you partial credit.
      1) (3 points) Size of the cache (in data bytes):
      2) (3 points) Number of lines per set:
      3) (3 points) Number of sets in the cache:
      4) (3 points) Size of the tag in bits:
      5) (3 points) Address format:
      6) (6 points) We are trying to read a byte with address 0xD2F456.
          What set number(s) (in hex) would we look at?
          If the byte is the cache, what is the tag (in hex) of its line?

6. (10 points) A processor has an 16-bit data bus with the read bus timing shown below. Assuming a processor clock rate of 8 MHz, what is the maximum data transfer rate?
7. (15 points) What is the maximum clock frequency possible for the following circuit? (In other words, what is the maximum clock frequency that will still guarantee correct behavior?) Use the following delay values, and assume all input signals become valid at time 0:

- AND: 4ns
- NAND: 3ns
- NOT: 2ns
- MUX: 5ns
- Tprop (TFF): 7ns
- Tsetup (TFF): 3ns
- Thold (TFF): 1ns
- Tprop (DFF): 8ns
- Tsetup (DFF): 3ns
- Thold (DFF): 1ns

**Note:** You must show the path in order to get credit.
8. (15 points) Given the following program fragment, only 4 data registers in the CPU, and 32-byte data cache with 4-byte memory blocks. Thus the cache has eight lines. Assume addition is done from left to right in the CPU. Assume all communication with RAM must be through the cache, and cache misses take quite a while. Assume LRU replacement for two-way set associative mappings.

Which of the two cache mapping functions (direct, or two-way set associative) would make the program for-loop fastest and why? Be sure to explain why the other mapping function would cause the program to be slower. I used four sentences to answer.

```c
int i, b[1020]; // 4-byte ints.
... // b is initialized
for(i = 0; i < 1000; i++)
  b[i] = b[i + 1] + b[i + 2] + b[i + 3] + b[i + 4];
```