INSTRUCTIONS

Answer questions 1 through 4 for the Breadth Exam and answer questions 1 through 7 for the Depth Exam. We RECOMMEND that students taking the Depth Exam spend the first two hours on questions 1 through 4, and the remaining two hours on questions 5 through 7. We have tried to make the questions quite specific. If, however, some confusion should arise, be sure to state all your assumptions explicitly.

BREADTH EXAM

1. Disk I/O Performance

If we are to design balanced computer systems, that is, computer systems with no performance bottlenecks, it is fairly obvious that the performance of the I/O system must be increased proportionately as CPU performance increases. An important component of I/O is disk I/O, and there are two aspects of disk I/O performance: *latency* and *bandwidth*.

(a) Discuss several ways of improving the latency of disk I/O, listing the pros, cons, and limits of each approach.

(b) Discuss several ways of improving the bandwidth of disk I/O, listing the pros, cons, and limits of each approach.

2. Conditional Branch Logic

Some instruction set architectures have conditional branch instructions that compare (e.g, with `=` and `<`) a value with zero to determine whether a branch is taken. Others have branch instructions that compare two values.

Below you will be asked to implement conditional branch logic. Assume that you are given gates (NOT, AND, OR, XOR, etc.) and a one-bit full adder with three inputs (`x`, `y`, `carry-in`) and two outputs (`sum` and `carry-out`). For simplicity, assume that values are 4-bit two's-complement numbers, do *not* use carry-lookahead, and concentrate on clarity and correctness.

(a) Implement the combinational logic for testing whether a value is either equal to zero or less than zero. Inputs: `test` and 4-bit `value`. Output: `branch`. Function (in the C language):

```
if (test==1) {
    branch = (value==0);
}
else {
    branch = (value<0);
}
```

(b) Implement the combinational logic for testing whether a value is either equal to or less than a second value. Inputs: `test`, 4-bit `value1`, and 4-bit `value2`. Output: `branch`. Function:

```
if (test==1) {
    branch = (value1==value2);
}
else {
    branch = (value1<value2);
}
```

(c) Give the advantages and disadvantages of the two types of branch instructions.

3. Bus Arbitration

Consider the problem of arbitrating for a common bus. There are three common strategies for coordinating access to this shared resource:
1) Daisy chain
2) Central or star arbiter
3) Taub's distributed scheme (hint: it uses wired-OR id lines lines).

Part 1: Design

Assume the bus can support 16 "widgets" (i.e., processors or other devices). Describe each of these three strategies, and the way processors are interconnected. A picture showing three widgets and the control signals that connect them across the bus is recommended.

Part 2: Trade-offs

Discuss the pros and cons of the different approaches. Specifically discuss issues related to
a) implementation cost (e.g., wires, gates, etc.)
b) performance
c) policy (priorities and fairness)
d) reliability

4. Virtual Memory

Virtual address translation is a mechanism that maps a virtual address into a physical address, thereby hiding details of a particular machine's underlying memory system from the user programmer.

Part 1: Page Tables

Describe the mapping of a 32-bit virtual address to a 36-bit physical address, using a multi-level pagetable. Assume that the page size is 4096 bytes, each level of the page table is a full page, and that the entire page table resides in physical memory. Sketch the page table and show which bits from the address are used to index into each level. Show the contents of a page table entry and explain how these are used to find the next level of the table. Assume you have one register containing the root or base address of the page table. Also assume that page table entries are stored in four bytes.

Part 2: Translation Lookaside Buffer

For the page table designed above, how many distinct page table accesses are required to complete a single data access (assume no special acceleration hardware)?

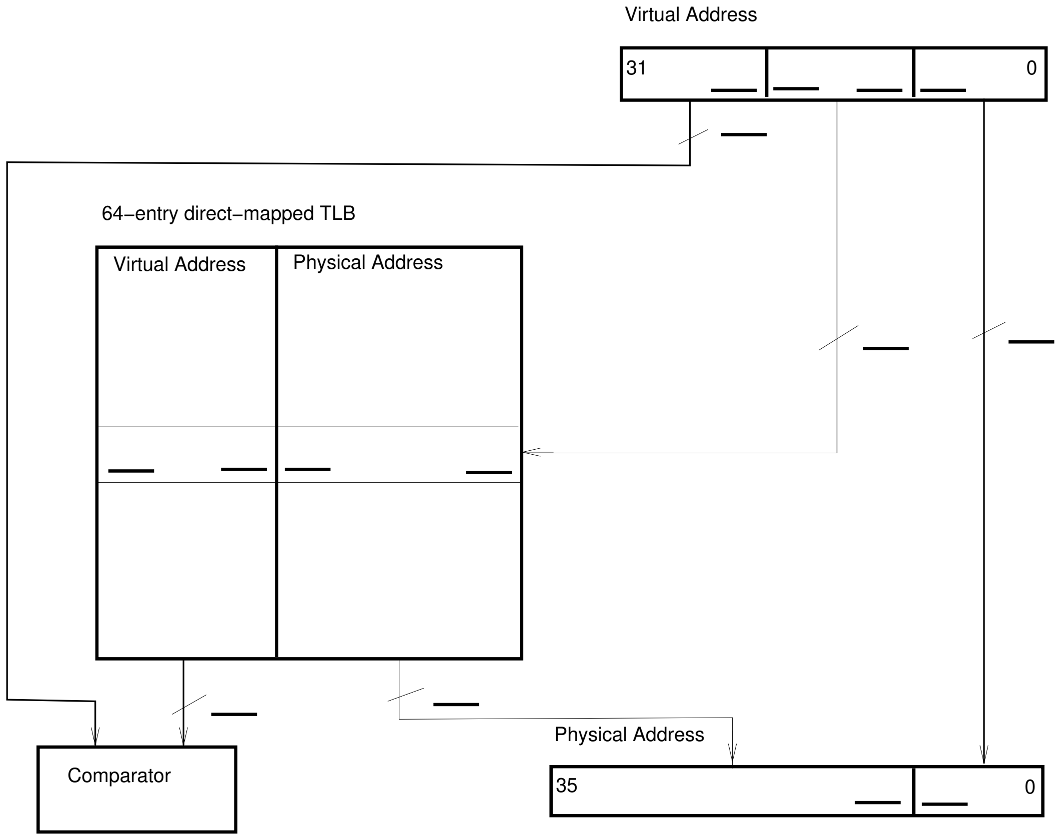
To reduce the average number of page table accesses, most systems provide a *translation lookaside buffer (TLB)*. Describe how a translation lookaside buffer works and what happens on both TLB hits and misses.

The picture on the next page shows a 64-entry direct-mapped TLB. Label the picture to show which bits are used to to index the TLB, which bits are involved in the comparison, and which bits are used to form the physical address. Label each of the signals (lines) to indicate how many bits are passed. For the fields in the TLB, show which bits of the virtual (physical) address are stored in each field.

Your number: ___________________

Be sure to write your numbers in the space provided.

HAND THIS PAGE IN WITH YOUR ANSWER BOOK!!!



DEPTH EXAM

Answer questions 5 through 7 for the Depth Exam (in addition to questions 1 through 4 which you should have already answered). The questions are quite specific. If, however, some confusion should arise, be sure to state all your assumptions explicitly.

5. Token Matching in Dataflow Machines

A key attribute of a dataflow machine is how a token (an operand in conventional terms) locates its partner token (other operand) so that, if both are available, a node (instruction) can fire (execute). If the partner token is not available, the current token must be saved so that it is found later by the partner.

(a) Describe how the above function is performed in a conventional dataflow machines (e.g., MIT Tagged-Token Dataflow Architecture or Manchester Dataflow Machine).

(b) Describe how the above function is performed in a machines that implements an *explicit token store (ETS)* (e.g., MIT Monsoon).

(c) Compare and contrast the above two approaches.

6. Instruction Units for Instruction-Level Parallelism

Recently computer designers have been implementing uniprocessors that execute multiple operations in parallel. Approaches include *superscalar*, *vector*, and *very long instruction word (VLIW).*

(a) Describe the above three approaches.

(b) Compare and contrast the above three approaches with respect to demands on instruction fetch logic.

(c) Compare and contrast the above three approaches with respect to demands on instruction decode logic.

7. Memory System Latency and Bandwidth

There are two main performance attributes of a memory system: *latency* and *bandwidth*. Some argue that reducing latency should be the primary consideration; increasing bandwidth is a secondary consideration. Others argue that increasing bandwidth should the primary consideration.

(a) Argue why reducing latency should be the primary consideration when designing a memory system.

(b) Argue why improving bandwidth should be the primary consideration when designing a memory system.