

Points missed: \_\_\_\_\_

Student's Name: \_\_\_\_\_

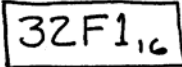
Total score: \_\_\_\_\_/100 points

East Tennessee State University  
Department of Computer and Information Sciences  
CSCI 4717 – Computer Architecture  
TEST 3 for Fall Semester, 2005

Section 201

**Read this before starting!**

- The total possible score for this test is 100 points.
- This test is closed book and closed notes
- You may use one sheet of scrap paper that you will turn in with your test.
- **When possible, indicate final answers by drawing a box around them. This is to aid the grader (who might not be me!) Failure to do so might result in no credit for answer.**  
**Example:**

A handwritten example of a boxed answer, showing the text "32F1" followed by a subscript "16" in a small circle, all enclosed in a hand-drawn rectangular box.

- **1 point will be deducted** per answer for missing or incorrect units when required. **No** assumptions will be made for hexadecimal versus decimal, so you should always include the base in your answer.
- If you perform work on the back of a page in this test, indicate that you have done so in case the need arises for partial credit to be determined.

“Fine print”

Academic Misconduct:

Section 5.7 "Academic Misconduct" of the East Tennessee State University Faculty Handbook, October 21, 2005:

"Academic misconduct will be subject to disciplinary action. Any act of dishonesty in academic work constitutes academic misconduct. This includes plagiarizing, the changing of falsifying of any academic documents or materials, cheating, and the giving or receiving of unauthorized aid in tests, examinations, or other assigned school work. Penalties for academic misconduct will vary with the seriousness of the offense and may include, but are not limited to: a grade of 'F' on the work in question, a grade of 'F' of the course, reprimand, probation, suspension, and expulsion. For a second academic offense the penalty is permanent expulsion."

### *Memory Management*

1. What does a “page fault” mean in terms of virtual memory? (3 points)
2. How does virtual memory allow programs that are larger than real memory to be run? (2 points)
3. What problem is typically caused by very small virtual memory page sizes? (2 points)
4. What problem is typically caused by large virtual memory page sizes? (2 points)

5. Using the page table shown to the right representing a specific process, calculate the physical address from the logical address  $235_{16}$ . Assume a page size of  $2^8 = 256$ . Be sure to show your work. (4 points)

Start of  
page table

23
6A
F4
36
3B
24
BB
...

6. If 6 processes are running in paged memory with a page size of 512 bytes, what is the largest amount of memory that is being wasted? (2 points)
7. True or false: The benefit of an inverted page table is that instead of maintaining an entry for every page in a process, it maintains an entry for every frame available in real memory. (2 points)

### *Assembly Language Details*

8. Assume the address field of an instruction contains a decimal 105. The address field represents something different depending on the addressing mode used by the instruction. For each of the addressing modes below, describe where the processor is getting its data from. (2 points each)
  - a.) Immediate:
  - b.) Indirect:

9. Store the value  $ABCDEF_{16}$  as individual bytes using big endian starting at address 100 in the figure to the right. Note that the figure will have blank locations after you are completed. (2 points)

$00FE_{16}$	
$00FF_{16}$	
$0100_{16}$	
$0101_{16}$	
$0102_{16}$	

10. The code below uses a three-operand instruction. In the space below, write three short programs that do exactly the same thing, one with two-operand instructions, one with one-operand instructions, and one with zero-operand instructions. If it fits the need of the instruction, feel free to use register names R1, R2, etc. (5 points)

ADD C, A, B ; C = A + B

<i>Two operand instructions</i>	<i>One operand instructions</i>	<i>Zero operand instructions</i>

*For the next 2 questions, use the following abbreviations for the stages of a 6-stage pipeline*

<i>Fetch instruction (FI)</i>	<i>Decode instruction (DI)</i>	<i>Calculate operands (CO)</i>
<i>Fetch operands (FO)</i>	<i>Execute Instruction (EI)</i>	<i>Write Operand (WO)</i>

11. List all of the stages such that any 2 of them occurring at the same time could result in a bus resource conflict? (2 points)
12. List all of the stages that are *always* used in the execution of an instruction. (Hint: Think of the simplest instruction and decide which stages are needed to execute it.) (2 points)

### *Pipelines and Branch Prediction*

13. In an ideal implementation, what is the speed up of a processor with a  $k$ -stage pipeline over a non-pipelined processor if the duration of each stage is  $\tau$ ? Don't consider pipeline flushes or delays incurred between stages. (2 points)
- a.)  $k \cdot \tau$     b.)  $k-2$     c.)  $k-1$     d.)  $k$     e.)  $(k+1) \cdot \tau$     f.)  $k+\tau-1$     g.) none of the above
14. How many bits are required for each conditional branch in a branch history table in order to remember the past 3 branch outcomes for a specific instruction? (2 points)
- a.) 2    b.) 3    c.) 4    d.) 5    e.) 6    f.) 7    g.) 8
15. In what level of the memory hierarchy are the branch history bits typically stored? (2 points)

16. Identify the two problems with having two pipelines (one for the branch stream and one for the non-branch stream) in order to minimize the delay caused by an incorrectly predicted branch. (3 points)
17. True or false: As far as the execution of a for-loop is concerned, the benefits of a loop buffer are only realized if the entire code executed for a for-loop can be contained in the buffer. (2 points)
18. How many cycles does prefetching the branch target save over not prefetching the branch target when a branch is incorrectly predicted? (2 points)
- a.) 0          b.) 1          c.) 2          d.) 3          e.) none of the above

*For problems 20, 21, and 22, consider the following section of code.*

```
for (i=0; i<10; i++)
{
    for (j=0; j<5; j++)
    {
        <code containing no conditional jumps>
    }
}
```

19. Once compiled, how many conditional jumps would be contained in the machine code resulting from the above section of code? (2 points)
20. After fully executing the above section of code, how many conditional jumps would the CPU have encountered? (2 points)
21. Using the static branch prediction algorithm “branch always,” how many of the conditional jumps calculated in the previous problem would have been predicted *incorrectly*? (2 points)
22. What are the three items that should be stored in a branch history table? (3 points)

### ***RISC Processors***

23. True or false: The purpose behind the delayed branch is to avoid flushing the pipeline. (2 points)

24. Place a check mark next to each statement below that *tends to be true* for a RISC processor. (1 point each)

- RISC processors have more specialized registers rather than general purpose.
- RISC processors have a very limited number of addressing modes.
- RISC processors using 2-way pipelined timing (only two instructions can be in the pipe at any on time) do so in order to avoid two simultaneous memory accesses.
- RISC processors use a fixed instruction length.

25. In order to avoid flushing the pipeline of a RISC processor, we need to modify the code to the right. If we wish to create an optimized delayed branch, which lines can be moved after L4? (2 points)

```
L1:  mov  a1,b1
L2:  add  ch,23
L3:  cmp  b1,ch
L4:  jne  label_a
L5:  inc  b1
```

- a.) L1 only    b.) L2 only    c.) Either L1 or L2    d.) Neither

26. Find the absolute minimum number of registers required to execute the code below assuming that every variable in the code will need to be placed in a register. (In other words, don't try to optimize the code, just the use of registers.) (2 points)

Answer: \_\_\_\_\_

```
int var0 = 50;
int var1 << cin;          // User input initializes var1
for (int i = 0; i < var1; i++)
{
    int var2 = i % 5;
    if(var2 == 0)
        for (int j=0; j<10; j++) var0++;
    else if (var2 <3)
        for (int j=0; j<10; j++) var0+=2;
    else
    {
        int var3 = i * 10;
        for (int j=0; j<var3; j++) var0--;
    }
}
```

### *Superscalar*

27. Which *two* of the following dependencies exhibit similar behavior? (2 points)

- a.) Data dependency            b.) Procedural dependency            c.) Resource conflict

28. Identify the write-read, write-write, and read-write dependencies in the instruction sequence below by entering each line pair with a dependency in the correct column of the table to the right. For example, if L1 and L4 had a write-write dependency (which they don't), you would enter L1-L4 in the column labeled "write-write". (4 points)

L1: R1 = R2 + R5  
 L2: R2 = R2 + 1  
 L3: R3 = R2 + R5  
 L4: R5 = R1 - R2  
 L5: R1 = R2 + 30

write-read	write-write	read-write

29. In the code below, identify references to initial register values by adding the subscript 'a' to the register reference. Identify new allocations to registers with the next highest subscript and identify references to these new allocations using the same subscript. (3 points)

R1 = R2 + R5  
 R2 = R2 + 1  
 R3 = R2 + R5  
 R3 = R3 + R2  
 R5 = R1 ÷ R3  
 R1 = R2 × R5

**Questions 32 through 35 are based on the "in-order issue/in-order completion" execution sequence shown in the figure to the right.**

Decode		Execute		Write		Cycle
I1	I2					1
I3	I4	I1		I2		2
I3	I4	I1				3
I5	I6		I3	I4		4
I5	I6		I3			5
	I6		I5			6
			I6			7
				I5	I6	8

30. Why do I3 and I4 need to stay in the decode stage of the pipeline for two cycles? (2 points)

31. Why doesn't I5 get written during cycle 7 instead of cycle 8? (2 points)

32. Why does I6 stay in the decode stage of the pipeline for one more cycle than I5? (2 points)

33. Which instructions could have been written earlier if this had been an "out-of-order completion" machine. (2 points)

### *SMP and Clusters*

34. Which cache coherence protocol uses distributed control, the directory protocol or snoopy protocol? (2 points)
35. Assume a multiprocessor system uses the MESI protocol. If the current state of a line in processor A's cache is exclusive and processor B loads the same line into its cache, what does the state of that line in processor A's cache change to? (2 points)  
 a.) modified    b.) exclusive    c.) shared    d.) invalid    e.) cannot be determined
36. Assume a multiprocessor system uses the MESI protocol. If the current state of a line in processor A's cache is exclusive and processor A modifies that data, but *does not* write the new value to main memory, what does the state of that line in processor A's cache change to? (2 points)  
 a.) modified    b.) exclusive    c.) shared    d.) invalid    e.) cannot be determined
37. Assume a multiprocessor system uses the MESI protocol. If the current state of a line in processor A's cache is shared and processor B modifies that data without updating main memory, what does the state of that line in processor A's cache change to? (2 points)  
 a.) modified    b.) exclusive    c.) shared    d.) invalid    e.) cannot be determined
38. Which SMP bus configuration has the most complex printed circuit board? (2 points)  
 a.) time-shared bus                                    b.) multiport memory                                    c.) central controller
39. The snoopy protocol is more suited to the \_\_\_\_\_ interconnection method for symmetric multiprocessors. (2 points)  
 a.) time-shared bus                                    b.) multiport memory                                    c.) central controller

*For the next 6 questions, identify whether the statement more closely identifies an SMP system or a cluster. (1 point each)*

- |     | SMP                      | Cluster                  |  |
|-----|--------------------------|--------------------------|--|
| 40. | <input type="checkbox"/> | <input type="checkbox"/> | Processors are loosely coupled – communication usually at the file level |
| 41. | <input type="checkbox"/> | <input type="checkbox"/> | Operation is closer to that of a single processor system                 |
| 42. | <input type="checkbox"/> | <input type="checkbox"/> | The newer of the two architectures                                       |
| 43. | <input type="checkbox"/> | <input type="checkbox"/> | Easier to build, especially by end-user                                  |
| 44. | <input type="checkbox"/> | <input type="checkbox"/> | Takes less physical space  |
| 45. | <input type="checkbox"/> | <input type="checkbox"/> | Superior scalability   |