Points missed: _____ Student's Name: ___________________________________

Total score: _____ /100 points

East Tennessee State University
Department of Computer and Information Sciences
CSCI 2150 (Tarnoff) – Computer Organization
TEST 3 for Fall Semester, 2004

Read this before starting!

- The total possible score for this test is 100 points.
- This test is closed book and closed notes.
- **All** answers **must** be placed in space provided. Failure to do so may result in loss of points.
- **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.
- ***Calculators are not allowed.*** Use the tables below for any conversions you may need. Leaving numeric equations is fine too.
- The table of assembly language commands is on the last page of this test. Remove it if you wish to have better access to it. Use the backside of it as your scrap paper. Turn it in with your test.

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binary	Hex
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Power of 2	Equals
2^3	8
2^4	16
2^5	32
2^6	64
2^7	128
2^8	256
2^9	512
2^{10}	1K
2^{20}	1M
2^{30}	1G

"Fine print"

Academic Misconduct:

Section 5.7 "Academic Misconduct" of the East Tennessee State University Faculty Handbook, June 1, 2001:

"Academic misconduct will be subject to disciplinary action. Any act of dishonesty in academic work constitutes academic misconduct. This includes plagiarism, 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."

1. How many latches does a 256 Meg SRAM with 8 data bits per location require? Leave your answer in the form of an equation with numeric values. (2 points)

   There are 256 Meg locations in a 256 Meg SRAM. (256×1 Meg = $2^8 \times 2^{20} = 2^{28}$) Since there are 8 data bits per location, a 256 Meg SRAM contains $2^{28} \times 8 = 2^{31}$ = 2,147,483,648 latches. (Note: you are not responsible for any of the mathematical calculations. Simply putting 256 Meg $\times$ 8 would have been sufficient.)

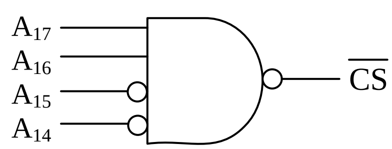
2. Circle ***all*** that apply. A storage cell in a DRAM: (4 points)

   a.) is volatile
   b.) is a capacitor
   c.) is cheaper than a cell in an SRAM
   d.) is a latch
   e.) must be refreshed regularly
   f.) is smaller than a cell in an SRAM
   g.) is typically used for cache RAM
   h.) is faster than a cell in an SRAM

3. Match each of the settings of the bus control signals $\overline{R}$ and $\overline{W}$ on the left with the bus operation on the right. (4 points)

$\overline{R}$	$\overline{W}$	Operation of the bus
0	0	Processor reads from memory
0	1	Illegal setting
1	0	The bus is idle
1	1	Processor writes to memory

4. What are the high and low addresses ***(in hexadecimal)*** of the memory range defined with the chip select shown to the right? (6 points)



   There are 18 address lines. This is found by noting that the highest address line has a subscript of 17 and therefore, since we begin counting at 0, we know that there are 18 address lines. Looking at the inputs to the NAND gate, we see that to set ^CS to zero, their values must be: A_{17}=1, A_{16}=1, A_{15}=0, and A_{14}=0. Therefore, the address lines have the following values for the high and low address:

   Low address: 11 0000 0000 0000 0000_2 = 30000_{16}
   High address: 11 0011 1111 1111 1111_2 = $33FFF_{16}$

5. For the chip select in the previous problem, how big is the memory chip that uses this chip select? (3 points)

   There are 14 address lines that go to the address inputs of the memory chip. Therefore, there are 2^{14} possible addresses meaning that the memory chip has $2^{14} = 2^4 \times 2^{10}$ = 16K memory locations.

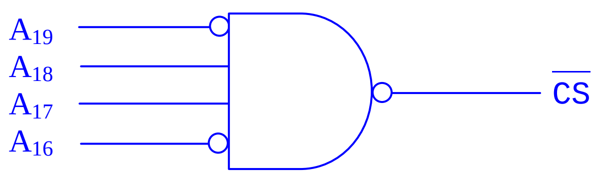
6. Using logic gates, design an active low chip select for a RAM placed in a 1 Meg memory space with a low address of 60000_{16} and a high address of $6FFFF_{16}$. ***Label all address lines used for chip select.*** (7 points)

   Since 1 Meg = 2^{20}, the processor must have 20 address lines coming out of it. (A_0 through A_{19})

   Converting the high and low addresses shows us where to draw the line separating the address lines that go to the chip select from the address lines that go to the memory chip.

   60000_{16} = 0110 0000 0000 0000 0000_2
   $6FFFF_{16}$ = 0110 1111 1111 1111 1111_2

   This shows that the lower sixteen address lines (A_0 through A_{15}) go to the memory chip, and the upper four address lines (A_{16} through A_{19}) go to the chip select. Also from this drawing, we see that $A_{19} = 0$, $A_{18} = 1$, $A_{17} = 1$, and $A_{16} = 0$. By inverting the inputs that are to be recognized as zeros, we get the following NAND circuit for the chip select:



7. What is the largest memory that can have a starting (lowest) address of $AC8000_{16}$? (3 points)

   Remember that the lowest address must have all zeros going to the address lines of the memory chip. Therefore, if we can determine the number of bits equal to zero starting with the least significant, A_0, and going left, then we know how many address lines can be used for the memory chip. Begin by converting $AC8000_{16}$ to binary.

   $AC8000_{16}$ = 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

   There are fifteen zeros before you get to the first 1 in the binary value of $AC8000_{16}$. Therefore, up to fifteen address lines can go to the memory chip. This gives us an answer of $2^{15} = 2^5 \times 2^{10}$ = 32K.

8. True or false: The address range $C000_{16}$ to $DFFF_{16}$ is a valid range for a single memory. (2 points)

   Begin by converting the low and the high addresses to binary.

	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
Low address =	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
High address =	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

   Note that a vertical division can be made between the binary values that remain constant (A_{13} through A_{15}) and the values that change from all zeros to all ones (A_0 through A_{12}). Therefore, this is a valid memory space for a single memory and the answer is **TRUE**.

9. True or false: There are more sectors per track in the outer tracks of a ***multiple zone recording*** hard drive configuration than there are in the inner tracks. (2 points)

   This is true. In order to take advantage of the tracks being longer on the outer edges, more sectors are added. There is not, however, a difference in the number of bytes per sector. That remains constant regardless of the track.

10. True or false: The Winchester-type head of a hard drive does not move from the landing zone until the hard drive platters are up to speed. (2 points)

    This is also true. The speed of the platters lifts the Winchester head off of the surface. The specified "fly height" can not be guaranteed until the platters are up to speed.

11. True or false: A small gap is left between the tracks of a hard drive disk in order to avoid data bleeding over into (interfering with) the data from other tracks. (2 points)

    This is also true. The gap between tracks is to protect data from neighboring tracks. The gap between sectors is to provide synchronization for the hard drive controller.

The table below represents a small section of a cache that uses direct mapping. Refer to it to answer questions 12, 13, and 14.

Line number (decimal & binary)	Tag (binary only)	Word within the block 000	001	010	011	100	101	110	111	
$99_{10} = 01100011_2$	011011010	00_{16}	61_{16}	$C2_{16}$	23_{16}	84_{16}	$E5_{16}$	46_{16}	$A7_{16}$	row a
$100_{10} = 01100100_2$	110011010	10_{16}	71_{16}	$D2_{16}$	33_{16}	94_{16}	$F5_{16}$	$\mathbf{56_{16}}$	$B7_{16}$	row b
$101_{10} = 01100101_2$	100110101	20_{16}	81_{16}	$E2_{16}$	43_{16}	$A4_{16}$	05_{16}	66_{16}	$C7_{16}$	row c
$102_{10} = 01100110_2$	101100111	30_{16}	91_{16}	$F2_{16}$	53_{16}	$B4_{16}$	15_{16}	76_{16}	$D7_{16}$	row d
$103_{10} = 01100111_2$	000011101	40_{16}	$A1_{16}$	02_{16}	63_{16}	$C4_{16}$	25_{16}	86_{16}	$E7_{16}$	row e
		col 0	col 1	col 2	col 3	col 4	col 5	col 6	col 7	

12. Assuming no leading zeros have been removed from any of the values shown in the table above, how many total lines does this cache have? (2 points)

    The direct mapping cache uses a portion of the block address to point to the line number of the cache where that line is to be stored. Therefore, all eight bits of the line number shown in the table must be used to identify a unique line. This means that there are $2^8 = 256$ lines in this cache. (Abnormally small.)

13. From what address in main memory did the value 56_{16} (the value in bold) come from? Leave your answer in binary. (3 points)

    A cache with 256 lines uses 8 bits to identify the line number. To address a word within a block of eight words, three bits are needed. The remaining bits are the tag. (We can assume that the tags in the table represent all of the bits, i.e., 9 bits.) Therefore, the figure below shows the allocation of the address bits to the different purposes of storing data within a cache.

9 bits	8 bits	3 bits
Tag	Line id	Word id

Now if we insert the tag, line id, and word id of the location holding 56_{16}, we get the following address.

9 bits	8 bits	3 bits
110011010	01100100	110

Converting this address to hex gives us:

$$1100\ 1101\ 0011\ 0010\ 0110 = CD326_{16}$$

14. A block containing the address $6533B_{16}$ is not contained in this cache. When loaded, which row (a through f) and column (0 through 7) will its value be stored in? (4 points)

Begin by converting this address to binary:

$$6533B_{16} = 0110\ 0101\ 0011\ 0011\ 1011$$

Now if we partition our address into the components shown in our table, we should see the different values assigned to each parameter.

9 bits	8 bits	3 bits
011001010	01100111	011

The column can be determined from the word id bits:

column 011 = column 3

For the row, simply determine which row corresponds to the line number 01100111:

$$2^6 + 2^5 + 2^2 + 2^1 + 2^0 = 64 + 32 + 4 + 2 + 1 = 103_{10}. = \text{row e}$$

15. True or false: A split cache system uses two caches, one for data and one for code. (2 points)

This is true. The other type of multiple cache systems uses levels, typically one inside the processor (Level 1) and one on the circuit board close to the processor (Level 2).

16. What is the purpose of pipelining? (3 points)

The purpose of pipelining is the improve the performance of the processor by making sure that no component is ever idle. If the hardware that fetches the next instruction can be fetching while the instruction decoder is decoding, then by all means do it. It will save time in the long run.

17. Assume a processor takes 3 cycles to execute any instruction (fetch, decode, execute)

a. How many cycles would a ***non-pipelined*** processor take to execute 6 instructions? (2 points)

A non-pipelined processor simply executes the instructions one at a time with no overlap. Therefore, the number of cycles equals 3 times the number of instructions:

$$\boxed{\text{number of cycles} = 3 \times 6 = 18 \text{ cycles}}$$

b. How many cycles would a ***pipelined*** processor take to execute 6 instructions? (2 points)

A pipelined processor overlaps 2 cycles for each instruction. Therefore, it will take 2 cycles to fill the pipeline, then one cycle per instruction to execute each one.

number of cycles = 2 + 6 = 8 cycles

Answer questions 18 through 23 using the following settings of the 8086 registers.

```
AX = 0180h        IP = 2122h        CS = 6000h
BX = AA55h        SP = 4344h        SS = 7000h
CX = 03C0h        DI = 6566h        DS = 8000h
DX = FFEEh        BP = 1234h        ES = 9000h
```

18. What is the value contained in the register BL? (2 points)

BL is the lower half (byte) of BX. Therefore, the value in BL is **55h**.

19. What is the physical address pointed to by ES:BP? (3 points)

ES contains the segment address and BP contains the pointer address. To figure out the physical address, begin by converting the 16-bit value in ES to the 20 segment address by adding a hex 0 to the end of the segment value (4 binary 0's).

ES = 9000h → the segment address is 9000**0**h (notice the added zero)

The pointer value (BP in this case) can then be added as an offset to the segment address.

```
 90000
+ 1234
-------
 91234
```

Therefore, the physical address pointed to by ES:BP (9000:1234) is **91234h**.

20. True or false: The physical address of the next instruction to be executed by the processor can be calculated from the above data? (2 points)

The answer is true. The physical address of the next instruction to execute is determined from the values contained in CS and IP (CS:IP). Since both of those values are present above (CS:IP = 6000:2122), then we can calculate the physical address of the next instruction to execute.

21. What is the value of SP after the execution of the instruction **`PUSH AX`**? (2 points)

If you go to the list of instructions on the last page, you'll see that executing a PUSH instruction "decrements SP by the size of the operand (two for 8 or 16 bit and four for 32 bit increments). Since AX is a 16 bit value, SP is decremented by 2 giving us 4344h – 2 = **4342h**.

22. Assume that the instruction **`INC DH`** is executed. How would the following flags be set? ***Write "N/A" if the flag was not affected.*** (3 points)

Referring to the list of instructions on the last page, we see that the INC adds 1 to the operand and affects the flags CF, AF, OF, PF, SF, and ZF. Since DH equals FF_{16} before the instruction is executed, it equals 00_{16} after the instruction is executed. Therefore, since it is equal to zero, the zero flag (ZF) is set to 1. Since zero is considered a positive value (the MSB of the result is 0), the sign flag (SF) is cleared to 0. Adding 1 to FF also generates a carry. Therefore, CF = 1.

ZF = __1__ CF = __1__ SF = __0__

23. Assume that the instruction **SAR BH,3** is executed. What would the new value of BH be? (3 points)

Referring to the list of instructions on the last page, we see that the SAR BH,3 will shift the 8-bit BH register 3 places to the right with the most significant bit duplicated to fill in the spaces left by the right shifts. In addition, the carry flag contains the last bit shifted out. So what does BH equal before the instruction?

BH = AAh = 10101010b

After the instruction, the three right most bits are shifted out, the last one, a zero, going into the carry flag (CF). The most significant bit, a one, is duplicated three times filling in from the left.

new BH = 11110101b = F5h

SAR modifies the flags CF, OF, PF, SF, and ZF. Since the new value of BH is not equal to zero, the zero flag (ZF) is cleared to 0. Since it is a negative value (the MSB of the result is set to 1), then the sign flag (SF) equals 1. Last of all, the last bit shifted out is in CF which is a zero.

ZF = __0__ CF = __0__ SF = __1__

24. Assume AX=1000h, BX=2000h, and CX=3000h. After the following code is executed, what would AX, BX, and CX contain? (3 points)

```
PUSH CX
PUSH BX
PUSH AX
POP CX
POP BX
POP AX
```

Place your answers in space below:

AX = **old CX = 3000h**

BX = **old BX = 2000h**

CX = **old AX = 1000h**

25. Which of the following best describes the operation of the instruction MOV AX, [1000h]? (2 pts)

a.) Load the 16-bit register AX with the number 1000_{16}.
b.) Store the value currently held in the 16-bit register AX to the address 1000_{16}.
c.) Load AX with the value stored at address 1000_{16}.
d.) Load AX with the value stored at address pointed to by the value stored at the address 1000_{16}.
e.) None of the above, this is an illegal instruction.

26. Of the following jump instructions, indicate which ones will jump to the address LOOP, which ones will simply execute the next address (i.e., not jump), and which ones you don't have enough information to tell.

Instruction		Current Flags	Jump to LOOP	Not jump to LOOP	Cannot be determined	
JNE	LOOP	SF=0, OF=1, CF=1	❑	❑	☒	(2 points)
JA	LOOP	CF=0, ZF=1, OF=0	❑	☒	❑	(2 points)
JNB	LOOP	SF=0, ZF=0, CF=0	☒	❑	❑	(2 points)
JNG	LOOP	ZF=1, SF=0, OF=0	☒	❑	❑	(2 points)

27. Name the two benefits of the segment/pointer addressing system of the 80x86. (3 points)

- It allows us to use 16 bit registers (word-length) to access larger (20-bit) address spaces
- It allows for relocatable code in memory

28. Using an original value of 10101010_2 and a mask of 00111100_2, calculate the results of a bitwise AND, a bitwise OR, and a bitwise XOR for these values. (2 points each)

Original value	Bitwise operation	Mask	Result
10101010_2	AND	00111100_2	**00101000_2**
10101010_2	OR	00111100_2	**10111110_2**
10101010_2	XOR	00111100_2	**10010110_2**

29. For each of the following binary bit patterns, set the parity bit for odd parity.

Odd parity sets the parity bit if the sum of the ones in the value is even and clears the parity bit if the sum of the ones in the value is odd.

Binary value								Parity	
0	0	1	0	1	1	0	1	**1**	(1 point)
1	1	1	1	0	1	0	0	**0**	(1 point)
1	0	1	0	1	0	1	0	**1**	(1 point)

30. Keeping your answer in decimal, calculate the basic checksum for the following sequence of bytes. (3 points)

10_{10} 20_{10} 5_{10} 1_{10} 2_{10}

The basic checksum is equal to the datasum with the carries out of the top most place thrown away. Therefore, the answer is $10 + 20 + 5 + 1 + 2 = 38_{10}$.

31. When using the 2's complement checksum, the sum of the datasum and the checksum should result in a binary value of what? (2 points)

Since the 2's complement checksum is equal to the negative of the datasum, then adding the checksum to the datasum will result in a **0**.

Name: ______________________________

DEC - Decrement
Usage: DEC   dest
Modifies flags: AF OF PF SF ZF
Description: Unsigned binary subtraction of one from the destination.

INC - Increment
Usage: INC   dest
Modifies flags: CF AF OF PF SF ZF
Description: Adds one to destination unsigned binary operand.

Jxx - Jump Instructions Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF=0 and ZF=0
JE	Jump if Equal	ZF=1
JG	Jump if Greater (signed)	ZF=0 and SF=OF
JGE	Jump if Greater or Equal (signed)	SF=OF
JL	Jump if Less (signed)	SF != OF
JMP	Unconditional Jump	unconditional
JNB	Jump if Not Below	CF=0
JNE	Jump if Not Equal	ZF=0
JNG	Jump if Not Greater (signed)	ZF=1 or SF != OF
JNL	Jump if Not Less (signed)	SF=OF
JZ	Jump if Zero	ZF=1

MOV - Move Byte or Word
Usage: MOV   dest,src
Modifies flags: None
Description: Copies byte or word from the "src" operand to the "dest" operand.

NOT - One's Compliment Negation (Logical NOT)
Usage: NOT   dest
Modifies flags: None
Description: Inverts the bits of the dest operand forming the 1s complement.

POP - Pop Word off Stack
Usage: POP   dest
Modifies flags: None
Description: Transfers word at the current stack top (SS:SP) to the destination then increments SP by two to point to the new stack top. CS is not a valid destination.

PUSH - Push Word onto Stack
Usage: PUSH   src
Modifies flags: None
Description: Decrements SP by the size of the operand (two for 8 or 16 bit and four for 32 bit, byte values are sign extended) and transfers one word from source to the stack top (SS:SP).

SAL/SHL - Shift Arithmetic Left / Shift Logical Left
Usage: SAL   dest,count   SHL   dest,count
Modifies flags: CF OF PF SF ZF (AF undefined)
Shifts the destination left by "count" bits with zeroes shifted in on right. The Carry Flag contains the last bit shifted out.

SAR - Shift Arithmetic Right
Usage: SAR   dest,count
Modifies flags: CF OF PF SF ZF (AF undefined)
Shifts the destination right by "count" bits with the current sign bit replicated in the leftmost bit. The Carry Flag contains the last bit shifted out.