1   True or False? (40 points.) This question occupies 3 pages.

1. On 16-bit machines, the largest memory size is 256 bytes.

   True   False

2. In both 32-bit machines and 64-bit machines, the char pointers have the same length (in number of bytes).

   True   False

3. The two's complement representation can encode the same number of negative integers and non-negative integers.

   True   False

4. After running the following code snippet on a big-endian machine, the short variable p will hold the value "0000".

```
1       char x = 0x82;
2       int y = (int)x;
3       short p = *(&y);
```

   True   False

5. A program compiled on one machine can still run on another machine that supports a different instruction set architecture.

   True   False

6. Executing the following two instructions sequentially (without anything happening in between) will not change the content in memory.

```
1       movq (%rax), %rdx
2       movq %rdx, (%rax)
```

   True   False

7. If after the following instructions **%rcx** holds the value of variable x, then **%rax** holds the value of &x.

```
1       movq (%rax), %rdx
2       movq (%rdx), %rcx
```

True   False

8. To implement "for" loops, one must use both conditional jumps and unconditional jumps.

True   False

9. Conditional jump instructions (jl, jne, etc.) use values in the general-purpose registers, in addition to the condition flags.

True   False

10. By convention, right before a callee function returns (before executing ret instruction), the stack pointer must always point to the top of the caller frame.

True   False

11. By convention, when a function returns, the callee function must explicitly let the caller function know which register holds the returned value.

True   False

12. By convention, if a caller function saves the value "5" in %rdi (a caller-saved register) right before entering a callee function, it can safely assume that when the callee returns, %rdi still holds the value "5".

True   False

13. If a C struct and a C union have the same set of member fields, the size of the struct (in bytes) must be greater than or equal to the size of the union.

True   False

14. On a 64-bit machine, if we define the following struct, then after the struct is initialized in memory, the address of the short field is exactly 8 bytes away from the address of the long field.

```
1    struct S {
2      long a;
3      short b;
4    } *p
```

True   False

15. Some programs exhbit locality, and some don't.

True   False

16. Comparing a 6-way set associative cache to a 4-way set associative cache with the same capacity and block size, the 6-way cache generally results in less conflict misses, but it is more difficult to implement.

True   False

17. Miss rate + hit rate is always 100% with respect to a single cache and a single program.

True   False

18. nuX86_64 is exactly the same as x86_64, except that it also defines a software-managed cache and several new instructions to manage this cache. This new cache is a microarchitectural feature, and not an architectural (ISA) feature, like all other caches.

True   False

19. Consider a program that accesses every 4-byte element of an integer array in order, and then it accesses every element of the same array in order again for a second time. The machine implements a fully-associative cache with a block size of 8 bytes. The miss rate of the program on this machine is 25%.

True   False

20. In general, for caches, the write back policy generates less traffic to memory than the write through policy.

True   False

2 (8 points.)

Assume that the length of `int` and `unsigned int` is 32 bits, signed integers use two's complement, and right/left shifts on signed integers are arithmetic. The following code generates arbitrary values for `int x` and `int y`, then converts them to `unsigned int`.

```
1  int x = random();       /* Create some arbitrary values */
2  int y = random();
3  unsigned int ux = x;    /* Convert to unsigned */
4  unsigned int uy = y;
```

Each part below is either a `C` expression of equality ("==") or a logical implication between two expressions ($A \implies B$). A non-zero value is true in `C`; zero means false.

Answer whether the expression or implication is *always* true for all possible x and y (type "Yes"), or not (type "No"). **If "No", give specific values** for the variables involved to make it false. Specific values must be in hex (starting with "0x"), except that you may also use TMin_{32} and TMax_{32} for the smallest and largest values (respectively) that can be stored in an `int`.

A. `(x > 0) && (y < 0)` $\implies$ `(x-y > 0)`

  Yes   No

B. `x > 11` $\implies$ `(x>>2) == (int)(ux>>2)`

  Yes   No

C. `x+0u >= 0u`

  Yes   No

D. `(int)(ux + 5*y) == x + 5*y`

  Yes   No

3 (16 points)

Consider this pseudocode compiled on a machine for which the sizes (in bytes) of `char`, `short`, and `int` are 1, 2, and 4, respectively.

```
1  int main()
2  {
3      char c[4] = {0x12,0x34,0x56,0x78};
4      char c1 = *((char *)(&c[2]));
5      short s1 = *((short *)(&c[0]));
6      short s2 = *((short *)(&c[1]));
7      int i1 = *((int *)(&c[0]));
8  }
```

A. What will be the following values on a big-endian machine (heximal format, starting with 0x)?

c1: _____

s1: _____

s2: _____

i1: _____

B. What will be the following values on a little-endian machine (heximal format, starting with 0x)?

c1: _____

s1: _____

s2: _____

i1: _____

4 (8 points.)

The following values are stored at the indicated memory addresses and registers:

Address	Value	Register	Value
0x210	0xAB	%rax	0x210
0x218	0xBC	%rcx	0x220
0x220	0x09	%rdx	0x1
0x228	0x42	%rsi	0x2
0x230	0x54	%rdi	0x4

Assume that the values in the "Value" column occupy eight bytes of storage. For each part (A-D) of this question, answer the value stored in %r8, in heximal format (starting with 0x):

Part A:

```
1  movq (%rax), %r8
2  addq 0x6(%rax, %rsi), %r8
```

Value stored in **%r8** after the **1st** line is _____

Value stored in **%r8** after the **2nd** line is _____

Part B:

```
1  movq 0x8(%rax), %r8
2  movq 0x6(%rcx, %rdx, 2), %r9
3  subq %r9, %r8
```

Value stored in **%r8** after the **1st** line is _____

Value stored in **%r8** after the **3rd** line is _____

Part C:

```
1  movq 0x228, %r8
2  leaq (%r8, %r8, 1), %r8
3  shrq %rdx, %r8
4  addq $0x228, %r8
```

Value stored in **%r8** after the **2nd** line is _____

Value stored in **%r8** after the **4th** line is _____

Part D:

```
1  movq 0x230, %r8
2  movq $0x230, %r9
3  notq %r8
4  notq %r9
5  xorq %r9, %r8
```

Value stored in **%r8** after the **5th** line is _____

5 (14 points) This question occupies 2 pages.

5.1 The assembly code below was compiled from a C function whose prototype is given below.

```
long funcP(long r, s, t);
```

```
1  _funcQ: #long funcQ (long x, long y)
2      addq   $2, %rdi
3     imulq   $3, %rdi
4      movq %rsi, %rax
5      addq %rdi, %rax
6      ret
7
8  _funcP: #long funcP (long r, long s, long t)
9      movq %rdi, %r8
10     subq %rdx, %r8
11
12    testq  %r8, %r8
13      jle foo
14
15    callq _funcQ
16      jmp bar
17
18   foo:
19     movq %rsi, %rax
20     movq %rdx, %rsi
21     movq %rax, %rdx
22    callq _funcQ
23
24   bar:
25      ret
```

Consider the assembly code above, assume that for parts A-D, the inputs are
`r=2, s=3, t=5.`

A. How many times is funcQ called?

FuncQ is called _____ times.

B. What are argument values, x and y, to funcQ when funcQ is called for the first time?

x = _____, y = _____

C. What is the return value of funcQ when it is called for the first time?

Return value of funcQ is _____

D. What is the return value of funcP?

Return value of funcP is: _____

5.2 The assembly code below was compiled from a C function whose prototype is given below.

```
int funcP(long[][] x);
```

```
1  _funcP: #int funcP (long[][] x)
2     movq $0, %rax
3     movq $0, %r8
4
5  .loop:
6     movq %r8, %rdx
7    imulq $0x10, %rdx
8     movq (%rdi, %rdx), %r9
9     movq 0x8(%rdi, %rdx), %r10
10
11    comq %r10, %r9
12      jle foo
13    addq %r9, %rax
14      jmp bar
15
16 .foo:
17    addq %r10, %rax
18
19 .bar:
20    incq %r8
21
22    comq $2, %r8
23    jle loop
24    ret
```

E. Consider the assembly code above, what will be the returned value, if the input is
`long[3][2]x = {{1,6}, {2,3}, {5,4}}?`

Return value is: _____

6   (14 Points)

Suppose we have a system with the following properties:

- The memory is byte addressable.
- Memory accesses are to **1-byte words** (not to 4-byte words).
- Addresses are 12 bits wide.
- The cache has four sets ($S = 4$). Each set consists of two lines (two-way set associative, $E = 2$). Each line holds four bytes of data ($B = 4$).
- LRU replacement is used for the cache.

The contents of the cache are as follows, with all addresses, tags, and values given in hex:

Set index	Valid	Tag	Byte 0	Byte 1	Byte 2	Byte 3
0	1	00	40	41	42	43
	1	83	FE	97	CC	D0
1	1	00	44	45	46	47
	0	83	–	–	–	–
2	1	00	48	49	4A	48
	0	40	–	–	–	–
3	1	FF	9A	C0	03	FF
	0	54	–	–	–	–

Part A. For this configuration, which address bits are used for the tag, set index, and block offset? Indicate your answer by writing a string with "t" to indicate tag bits, "s" to indicate the index bits, and "b" to indicate block offset bits. For example, ttttsssbbbb.

Answer: _____

Part B. For each of the following memory reads which are **carried out in sequence** as listed, indicate two things:

1. Will it be a cache hit or miss?

2. If it is a hit, give the value of the corresponding memory location (in hex, starting with 0x). If it is a miss, write "unknown" in the "Value" column.

In total, there are 12 blanks to fill. The cache and memory are affected by nothing except the operations listed.

Address	Hit/Miss	Value
0x9		
0x54E		
0x836		
0xABD		
0x831		
0xFFF		