Midterm Exam

ⓘ This is a preview of the published version of the quiz

Started: Aug 24 at 9:16am

Quiz Instructions

Instructions

- This exam is an individual effort.
- You are not permitted to help others, in any way, with this exam.
- You are not permitted to release or to discuss this exam with anyone, except the course staff, until given permission to do so by the instructors (which will not occur until all students have completed the exam. There may be exceptional cases that take it late).
- A simple calculator is permitted, but won't prove to be helpful (we don't think).
- You have 180 minutes, from first exposure through submission to take this exam. Do not attempt to "peek", "check", or "test" the exam. This will start your clock.
- We only expect the exam to take 70-90 minutes.
- The exam counts for the 25% "exam portion" of the midterm grade, but is reduced to counting as a "double homework" for the final grade.
- In order to make the exam an "invested but low stakes" experience, half of this exam's weight toward the final grade may be dropped as one of the two "homework drops", but the full weight can't be dropped.
- Taking this exam closed book/closed notes is strongly recommended -- as nothing will be allowed during the final exam.

Question 1 15 pts

Integers (5 points, 1 point per blank)

Fill in the five empty boxes in the table below when possible and indicate "UNABLE" if impossible.

	4-bit 2s complement signed	**4-bit unsigned**

Binary representation of 5 decimal		----------
Binary representation of decimal -8		----------
Binary representation of decimal 8		---------- ----------
Binary representation of Tmin (negative)		----------
Integer (Decimal) value of (5 + 6)		----------

Question 2: Floats

This question is based upon an IEEE-like floating point format with the following specification:

- 10-bit width
- There is s = 1 sign bit
- There are k = 4 exponent bits
- Wherever rounding is necessary, round-to-even should be In addition, you should give the rounded value of the encoded floating point number.

Question 2 1 pts

Question 2: Floats

2(A) (1 points) What is the bias?

Question 3 1 pts

Question 2: Floats

2(B)(1 point) What is the exponent (actual exponent, not field value) for **de**normalized numbers?

Question 4 1 pts

Question 2: Floats

2(C) (1 points) What is the maximum exponent (actual exponent, not field value) for normalized numbers?

Question 5 1 pts

Question 2: Floats

2(D) (1 points) What exponent represents an infinity? Answer in binary.

Question 6 6 pts

Question 2: Floats

2(E-H) (6 points) Fill in the following:

Value	**Binary Representation**	**Rounded Value a reduced decimal fraction**

21/4		--
+/- Numerator/Denominator Sign: Numerator: / Denominator: Important notes: • Sign should be + or - • Fully reduce the fraction	1000010000	--

3. (20 points) **Assembly**

Please consider the following assembly code segment:

```
(gdb) (gdb) disassemble loop
Dump of assembler code for function loop:
   0x0000000000001169 <+0>:     endbr64
   0x000000000000116d <+4>:     push   %rbp
   0x000000000000116e <+5>:     mov    %rsp,%rbp
   0x0000000000001171 <+8>:     sub    $0x20,%rsp
   0x0000000000001175 <+12>:    mov    %edi,-0x14(%rbp)
   0x0000000000001178 <+15>:    mov    %esi,-0x18(%rbp)
   0x000000000000117b <+18>:    mov    -0x14(%rbp),%eax
   0x000000000000117e <+21>:    mov    %eax,-0x8(%rbp)
   0x0000000000001181 <+24>:    jmp    0x11cb <loop+98>
   0x0000000000001183 <+26>:    mov    -0x8(%rbp),%eax
   0x0000000000001186 <+29>:    mov    %eax,-0x4(%rbp)
   0x0000000000001189 <+32>:    jmp    0x1199 <loop+48>
   0x000000000000118b <+34>:    mov    $0x58,%edi               # 0x58 is "X"
   0x0000000000001190 <+39>:    call   0x1060 <putchar@plt>
   0x0000000000001195 <+44>:    subl   $0x1,-0x4(%rbp)
   0x0000000000001199 <+48>:    cmpl   $0x0,-0x4(%rbp)
   0x000000000000119d <+52>:    jg     0x118b <loop+34>
   0x000000000000119f <+54>:    mov    0x2e6a(%rip),%rax
   0x00000000000011a6 <+61>:    mov    %rax,%rdi
   0x00000000000011a9 <+64>:    call   0x1070 <fflush@plt>
   0x00000000000011ae <+69>:    mov    $0xa,%edi                # 0xa is '\n'
```

```
0x00000000000011b3 <+74>:    call   0x1060 <putchar@plt>
0x00000000000011b8 <+79>:    mov    0x2e51(%rip),%rax
0x00000000000011bf <+86>:    mov    %rax,%rdi
0x00000000000011c2 <+89>:    call   0x1070 <fflush@plt>
0x00000000000011c7 <+94>:    subl   $0x1,-0x8(%rbp)
0x00000000000011cb <+98>:    mov    -0x8(%rbp),%eax
0x00000000000011ce <+101>:   cmp    -0x18(%rbp),%eax
0x00000000000011d1 <+104>:   jg     0x1183 <loop+26>
0x00000000000011d3 <+106>:   nop
0x00000000000011d4 <+107>:   nop
0x00000000000011d5 <+108>:   leave
0x00000000000011d6 <+109>:   ret
```

Question 7 4 pts

3(A) (4 points): How many loops are within this question?

Question 8 4 pts

3(B) (4 points): How many if statements are within this question (that can't be considered part of the pre-test for a while or for loop)?

Question 9 4 pts

3(C) (4 points): Do two or more loops share the same loop control variable (a variable which is updated by the body of the loop and used as part of the test for the loop)?

- Yes
- No

Question 10 4 pts

3(D) (4 points): Do two or more loops share the same end point? In other words, do they stop when the loop control variable reaches the same value or condition?

- Yes

○
No

Question 11 4 pts

3(E) (4 points): If the function is called as "loop(10,5)", how many lines of output are produced?

4. (20 points) **Structs and Alignment**

Consider the following struct:

```
struct {
  int i;         // ints are a 4-byte type
  short sa[4];   // shorts are a 2-byte type
  long l;        // longs are an 8-byte type
  char c;        // chars are a 1-byte type
} exam;
```

Assume a system which requires “natural alignment”, i.e. each type needs to be aligned to a multiple of its size (width).

Question 12 3 pts

4(A) (3 points): How many bytes of padding would the compiler place immediately after `int i`?

Question 13 4 pts

4(B) (4 points): How many bytes of padding would the compiler place immediately after `short sa[4]`?

Question 14 4 pts

4(C) (4 points): How many bytes of padding would the compiler place immediately after `long l`?

Question 15 3 pts

4(D) (3 points): (3 points) How many bytes of padding would the compiler place immediately after `char c`?

Question 16 3 pts

4(E) (3 points): How many bytes would be reported by "sizeof(struct exam)"

Question 17 3 pts

4(F) (3 points): At most, how many bytes could be saved by reordering the fields of the struct?

Question 18 3 pts

Arrays Sizes (4 points)

Consider the following definitions in an x86-64 system with 8-byte pointers and 4-byte ints. Answer with only a decimal number

Definition A
int numbersA[3][4][5]; // ints are 4 bytes

5(a)(1.5 point): How many bytes are allocated to numbersA? (Write “UNKNOWN” if not knowable):

Bytes

Hint: Think sizeof()

5(b) (1.5 point): How many bytes are allocated per row of numbersA? (Write “UNKNOWN” if not knowable): Bytes

Hint: Think sizeof()

Question 19 2 pts

Array Arithmetic

5(c) (2 points): Consider the following definitions as implemented on a shark machine, i.e. x86-64. What is the difference, i.e. number of bytes, between numbers[1][2] and numbers[2][0]?

```
int numbers[ 3 ][ 5 ];
```

6. Switch Statement (10 points)

Please consider the following assembly, compiled on a shark machine:

```
(gdb) disassemble foo
Dump of assembler code for function foo:
   0x00000000000011d0 <+0>:     endbr64
   0x00000000000011d4 <+4>:     push   %rbp
   0x00000000000011d5 <+5>:     mov    %edi,%ebp
   0x00000000000011d7 <+7>:     lea    0xe26(%rip),%rdi        # 0x2004
   0x00000000000011de <+14>:    push   %rbx
   0x00000000000011df <+15>:    mov    %esi,%ebx
   0x00000000000011e1 <+17>:    sub    $0x8,%rsp
   0x00000000000011e5 <+21>:    call   0x1070 <puts@plt>
   0x00000000000011ea <+26>:    mov    0x2e1f(%rip),%rdi        # 0x4010 <stdout@GLIBC_2.2.5>
   0x00000000000011f1 <+33>:    call   0x1090 <fflush@plt>
   0x00000000000011f6 <+38>:    lea    -0x3(%rbx),%esi
   0x00000000000011f9 <+41>:    cmp    $0x6,%esi
   0x00000000000011fc <+44>:    ja     0x1258 <foo+136>
```

```
   0x00000000000011fe <+46>:    lea    0xe13(%rip),%rdx        # 0x2018
   0x0000000000001205 <+53>:    movslq (%rdx,%rsi,4),%rax
   0x0000000000001209 <+57>:    add    %rdx,%rax
   0x000000000000120c <+60>:    notrack jmp *%rax
   0x000000000000120f <+63>:    nop
   0x0000000000001210 <+64>:    mov    %ebp,%eax
   0x0000000000001212 <+66>:    add    $0x8,%rsp
   0x0000000000001216 <+70>:    shr    $0x1f,%eax
   0x0000000000001219 <+73>:    pop    %rbx
   0x000000000000121a <+74>:    add    %ebp,%eax
   0x000000000000121c <+76>:    pop    %rbp
   0x000000000000121d <+77>:    sar    %eax
   0x000000000000121f <+79>:    ret
   0x0000000000001220 <+80>:    lea    0x0(,%rbp,8),%eax
   0x0000000000001227 <+87>:    sub    %ebp,%eax
   0x0000000000001229 <+89>:    mov    %eax,%ebp
   0x000000000000122b <+91>:    add    $0x8,%rsp
   0x000000000000122f <+95>:    lea    0x3(%rbp),%eax
   0x0000000000001232 <+98>:    pop    %rbx
   0x0000000000001233 <+99>:    pop    %rbp
   0x0000000000001234 <+100>:   ret
   0x0000000000001235 <+101>:   nopl   (%rax)
   0x0000000000001238 <+104>:   movslq %ebp,%rax
   0x000000000000123b <+107>:   add    $0x8,%rsp
   0x000000000000123f <+111>:   sar    $0x1f,%ebp
   0x0000000000001242 <+114>:   imul   $0x38e38e39,%rax,%rax
   0x0000000000001249 <+121>:   pop    %rbx
   0x000000000000124a <+122>:   sar    $0x21,%rax
   0x000000000000124e <+126>:   sub    %ebp,%eax
   0x0000000000001250 <+128>:   pop    %rbp
   0x0000000000001251 <+129>:   ret
   0x0000000000001252 <+130>:   nopw   0x0(%rax,%rax,1)
   0x0000000000001258 <+136>:   add    $0x8,%rsp
   0x000000000000125c <+140>:   lea    -0x1(%rbp),%eax
   0x000000000000125f <+143>:   pop    %rbx
   0x0000000000001260 <+144>:   pop    %rbp
   0x0000000000001261 <+145>:   ret
End of assembler dump.
```

And the following memory dump:

```
(gdb) x/16wd 0x2008
0x2008: 1952673397      544108393       560951142       0
0x2018: -3576   -3565   -3520   -3552
0x2028: -3592   -3520   -3592   990059265
0x2038: 56      6       -4116   108
```

Question 20 2 pts

At what address does the jump table start? [jmp_start]

Note: Answer in HEX, prefixing with 0x, and leaving off any leading 0s.

Question 21 2 pts

At what address does the code for the default case begin? [def_addr]

Note: Answer in HEX, prefixing with 0x, and leaving off any leading 0s.

Question 22 2 pts

Assume that this code is for a "switch (x)", what value of x is associated with the 0th entry of the jump table?

Question 23 2 pts

How many cases "fall through" to another case within the jump table?

Question 24 2 pts

Assume that this code is for a "switch (x)", what is the maximum value of x managed by the jump table?

Question 25 1 pts

Part 6(A): Caching

Given a model described as follows:

- Number of sets: 4
- Total size: 32 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

6(A)(1) (1 point) How many bits for the block offset?

Question 26 1 pts

Part 6(A)(1): Caching

Given a model described as follows:

- Number of sets: 4
- Total size: 32 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

6(A)(3) (1 point) How many bits for the set index?

Question 27 1 pts

Part 6(A)(2): Caching

Given a model described as follows:

- Number of sets: 4
- Total size: 32 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

6(A)(2) (1 point) How many bits for the tag?

Question 28 3 pts

6(A)(3) (3 points): Locality

Given a model described as follows:

- Number of sets: 4
- Total size: 32 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

What is the maximum stride (index step) size while sequentially accessing a 1D char array to maintain a cache miss rate of no more than 50%?

Question 29 12 pts

7(A)(4-9) Caching (12 points, 1 point each):

Given a model described as follows:

- Number of sets: 4
- Total size: 32 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

Consider the following memory access trace, which is in order and begins at the beginning of time. For each of the following memory accesses, please indicate if it hits or misses, and if it misses. In the event

of a miss, please indicate of the miss evicts another entry or allocates (makes use of) an unused one, or whether it is a capacity, conflict, or compulsory (cold) miss, as prompted.

Question Number	Address	Hit or Miss? Select one (per row):	Miss Type (Choose N/A for Hit)? Select one (per row)
	0x54		
7(A)(4)	0x50	[Select]	[Select]
7(A)(5)	0x56	[Select]	[Select]
	0x02		
7(A)(6)	0xA4	[Select]	[Select]
6(A)(7)	0x06	[Select]	[Select]
7(A)(8)	0xF4	[Select]	[Select]
7(A)(9)	0x57	[Select]	[Select]

Question 30 2 pts

8. (2 points): Memory Hierarchy and Effective Access Time

Imagine a system with a main memory layered beneath a cache:

- The cache has a 2ns access time.
- The main memory has an access time of 10ns.

- The cache miss rate is 10%.
- In the event of a miss, memory access time and cache access time do **not** overlap.

8(A) (2 points) What is the effective, overall access time in ns?

Not saved Submit Quiz