ACTUAL MIDTERM EXAM

This is a preview of the published version of the quiz

Started: Mar 10 at 12:30am

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.
- This exam is closed book/closed notes. This is to ensure the exam is a good simulation of the final exam
- This is a self-proctored exam: You are responsible for ensuring that you, yourself, follow the rules and have a good and proper exam experience.

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.

- When providing binary, write digits without any prefixes or spaces. A suffix if "b" is optional (still no spaces).

- When writing out hexadecimal, write digits without any spaces and prefix with 0x (lowercase x, and still no spaces).
- Include no extraneous characters. Canvas just string matches. It does not understand meaning.

	4-bit 2s complement signed	**4-bit unsigned**
Binary representation of 9 decimal		
Binary representation of decimal -8		----------
Integer (Decimal) value of most positive number		----------
Integer (Decimal) value of (6 + 3)		----------

Question 2: Floats

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

- 8-bit width

- There is s = 1 sign bit
- There are k = 3 fraction/mantissa bits
- Wherever rounding is necessary, round-to-even should be used. 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) Consider the number line represented by this format. What is the maximum distance between two adjacent denormalized numbers? Round to the nearest power of 2 (it won't be exactly half way) and give the exponent (in decimal).

Question 4 1 pts

Question 2: Floats

2(C) (1 points) Consider the number line represented by this format. What is the maximum distance between two adjacent normalized numbers? Round to the nearest power of 2 (it won't be exactly half way) and give the exponent (in decimal).

Question 5 1 pts

Question 2: Floats

2(D) (1 points) If you take the greatest magnitude positive number representable on this number line, multiply it by 2, and then subtract 5, what do you get? (Do NOT include the sign).

Question 6 6 pts

Question 2: Floats

2(E-H) (6 points) Fill in the following, use round-by-two, as necessary:

Value	**Binary Representation**	**Decimal value**
-54		
Rounding error for the above (Decimal difference: Original - Rounded)	----------	

3. (20 points) **Assembly**

Please consider the following assembly code segment:

```
(gdb) disassemble loop
Dump of assembler code for function loop:
   0x0000000000001180 <+0>:     endbr64
   0x0000000000001184 <+4>:     push   %r15
   0x0000000000001186 <+6>:     xor    %eax,%eax
   0x0000000000001188 <+8>:     mov    $0x2,%r15d
   0x000000000000118e <+14>:    push   %r14
   0x0000000000001190 <+16>:    mov    %edx,%r14d
   0x0000000000001193 <+19>:    push   %r13
   0x0000000000001195 <+21>:    mov    %edi,%r13d
   0x0000000000001198 <+24>:    push   %r12
   0x000000000000119a <+26>:    push   %rbp
   0x000000000000119b <+27>:    mov    %esi,%ebp
   0x000000000000119d <+29>:    push   %rbx
   0x000000000000119e <+30>:    sub    $0x8,%rsp
   0x00000000000011a2 <+34>:    cmp    %r14d,%eax
   0x00000000000011a5 <+37>:    jl     0x11c0 <loop+64>
   0x00000000000011a7 <+39>:    add    $0x8,%rsp
   0x00000000000011ab <+43>:    pop    %rbx
   0x00000000000011ac <+44>:    pop    %rbp
   0x00000000000011ad <+45>:    pop    %r12
   0x00000000000011af <+47>:    pop    %r13
   0x00000000000011b1 <+49>:    pop    %r14
   0x00000000000011b3 <+51>:    pop    %r15
   0x00000000000011b5 <+53>:    ret
   0x00000000000011b6 <+54>:    cs nopw 0x0(%rax,%rax,1)          # Safe to ignore this line
   0x00000000000011c0 <+64>:    xor    %r12d,%r12d
   0x00000000000011c3 <+67>:    test   %r13d,%r13d
   0x00000000000011c6 <+70>:    jle    0x1212 <loop+146>
   0x00000000000011c8 <+72>:    nopl   0x0(%rax,%rax,1)
   0x00000000000011d0 <+80>:    xor    %ebx,%ebx
   0x00000000000011d2 <+82>:    test   %ebp,%ebp
   0x00000000000011d4 <+84>:    jle    0x11f8 <loop+120>
   0x00000000000011d6 <+86>:    cs nopw 0x0(%rax,%rax,1)        # Safe to ignore this line
   0x00000000000011e0 <+96>:    mov    0x2e29(%rip),%rsi        # 0x4010 <stdout@GLIBC_2.2.5>
   0x00000000000011e7 <+103>:   mov    $0x6f,%edi
   0x00000000000011ec <+108>:   add    $0x1,%ebx
   0x00000000000011ef <+111>:   call   0x1050 <putc@plt>
   0x00000000000011f4 <+116>:   cmp    %ebx,%ebp
   0x00000000000011f6 <+118>:   jne    0x11e0 <loop+96>
   0x00000000000011f8 <+120>:   mov    0x2e11(%rip),%rsi        # 0x4010 <stdout@GLIBC_2.2.5>
   0x00000000000011ff <+127>:   mov    $0xa,%edi
   0x0000000000001204 <+132>:   add    $0x1,%r12d
   0x0000000000001208 <+136>:   call   0x1050 <putc@plt>
   0x000000000000120d <+141>:   cmp    %r12d,%r13d
   0x0000000000001210 <+144>:   jne    0x11d0 <loop+80>
   0x0000000000001212 <+146>:   mov    0x2df7(%rip),%rsi        # 0x4010 <stdout@GLIBC_2.2.5>
   0x0000000000001219 <+153>:   mov    $0xa,%edi
   0x000000000000121e <+158>:   call   0x1050 <putc@plt>
   0x0000000000001223 <+163>:   mov    $0x1,%eax
   0x0000000000001228 <+168>:   cmp    $0x1,%r15d
   0x000000000000122c <+172>:   je     0x11a7 <loop+39>
```

```
   0x0000000000001232 <+178>:   mov    $0x1,%r15d
   0x0000000000001238 <+184>:   jmp    0x11a2 <loop+34>
End of assembler dump.
(gdb)
```

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 initial conditions? In other words, are two or more loops, regardless of what is accomplished within the body of each loop, initialized to begin with the same relevant state, i.e. initial values for variables relevant to each iteration of the loop> (Answer "Yes", or "No")

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? (Answer "yes" or "no").

Yes

No

Question 11 4 pts

3(E) (4 points): If the function is called as "loop(4, 2, 1, 0, 0)", how many non-empty (more nothing or blank space) lines of output are produced? Please answer with a specific number in decimal or, if it isn't possible to know given the code provided, please answer with "UNKNOWN".

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

Consider the following struct:

```
struct {
  short s1[2];    // shorts are a 2-byte type
  long l;         // longs are an 8-byte type
  char c;         // chars are a 1-byte type
  short s2[4];    // shorts are a 2-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 4 pts

4(A) (4 points): How many bytes of padding would the compiler place immediately after `s1`?

Question 13 4 pts

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

Question 14 4 pts

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

Question 15 4 pts

4(D) (4 points): How many bytes would of padding would the compiler place immediately after `s2`?

Question 16 4 pts

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

Question 17 3 pts

Arrays Sizes (4 points)

Please consider the **original** `struct exam` from above. Answer with only a decimal number.

Definition A
struct exam e1[2][3][4];

5(a)(1.5 point): How many bytes are allocated to e1? (Write “UNKNOWN” if not knowable): Bytes

Hint: Think sizeof()

5(b) (1.5 point): What is the offset, in bytes, from the beginning of the array to e1[1][2][3].s2[1] Bytes

Question 18 2 pts

Array Arithmetic

5(c) (2 points): How many bytes are allocated for `arr` ? Assume the x86-64 shark machines.

```
short *arr = malloc (10*sizeof(short));
```

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:
```

```
   0x0000000000001200 <+0>:     endbr64
   0x0000000000001204 <+4>:     lea    -0x3(%rsi),%eax
   0x0000000000001207 <+7>:     cmp    $0x6,%eax
   0x000000000000120a <+10>:    ja     0x1233 <foo+51>
   0x000000000000120c <+12>:    lea    0xdf1(%rip),%rdx        # 0x2004 or 8196 decimal
   0x0000000000001213 <+19>:    movslq (%rdx,%rax,4),%rax
   0x0000000000001217 <+23>:    add    %rdx,%rax
   0x000000000000121a <+26>:    notrack jmp *%rax
   0x000000000000121d <+29>:    nopl   (%rax)
   0x0000000000001220 <+32>:    lea    (%rdi,%rdi,8),%edi
   0x0000000000001223 <+35>:    lea    -0x4(%rdi),%eax
   0x0000000000001226 <+38>:    ret
   0x0000000000001227 <+39>:    nopw   0x0(%rax,%rax,1)
   0x0000000000001230 <+48>:    add    $0x1,%edi
   0x0000000000001233 <+51>:    mov    %edi,%eax
   0x0000000000001235 <+53>:    sub    %esi,%eax
   0x0000000000001237 <+55>:    ret
   0x0000000000001238 <+56>:    nopl   0x0(%rax,%rax,1)
   0x0000000000001240 <+64>:    lea    (%rdi,%rdi,2),%eax
   0x0000000000001243 <+67>:    ret
   0x0000000000001244 <+68>:    nopl   0x0(%rax)
   0x0000000000001248 <+72>:    mov    %edi,%eax
   0x000000000000124a <+74>:    shr    $0x1f,%eax
   0x000000000000124d <+77>:    add    %eax,%edi
   0x000000000000124f <+79>:    mov    %edi,%eax
   0x0000000000001251 <+81>:    sar    %eax
   0x0000000000001253 <+83>:    ret
End of assembler dump.

(gdb) x/20ld 0x2000
0x2000: 131073  -3556   -3553   -3524
0x2010: -3516   -3537   -3540   -3540
```

```
0x2020: -3924   -3920   -3879   -3873
0x2030: -3884   -3888   -3888   680997
0x2040: 990059265       60      6       -4128
```

```
(gdb) x/20x 0x2000
0x2000: 0x00020001      0xfffff21c      0xfffff21f      0xfffff23c
0x2010: 0xffffff244     0xfffff22f      0xfffff22c      0xfffff22c
0x2020: 0xfffff0ac      0xfffff0b0      0xfffff0d9      0xfffff0df
0x2030: 0xfffff0d4      0xfffff0d0      0xfffff0d0      0x000a6425
0x2040: 0x3b031b01      0x0000003c      0x00000006      0xfffefe0
```

Question 19 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 20 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 21 2 pts

How many cases fall through to the next case after executing some of their own code?

Question 22 2 pts

How many values managed by the jump table are to code for the default case?

Question 23 2 pts

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

Question 24 1 pts

Part 6(A): Caching

Given a cache described as follows:

- Number of sets: 4
- Total size: 64 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 1-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

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

Question 25 1 pts

Part 6(A)(1): Caching

Given a cache described as follows:

- Number of sets: 4
- Total size: 64 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 1-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

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

Question 26 1 pts

Part 6(A)(2): Caching

Given a cache described as follows:

- Number of sets: 4
- Total size: 64 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 1-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

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

Question 27 1 pts

6(A)(3): Caching

Given a cache described as follows:

- Number of sets: 4

- Total size: 64 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 1-way set associative
- Replacement policy: Set-wise LRU
- 8-bit addresses

What is the maximum stride (index step) size while sequentially accessing a 1D long (8-byte type) array to maintain a cache miss rate of no more than 25%?

Question 28 14 pts

7(A-G) Caching (14 points, 2 point each):

Given a cache described as follows:

- Number of sets: 4
- Total size: 64 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 1-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)**
----------	0x4A	----------	----------
----------	0x9D	----------	----------

7(A)	0x4B	[Select]	[Select]
7(B)	0xEA	[Select]	----------
7(C)	0x3C	[Select]	[Select]
7(D)	0x70	[Select]	----------
7(E)	0x84	[Select]	----------
7(F)	0x42	[Select]	[Select]

Question 29 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 4ns access time.
- The main memory has an access time of 8ns.
- In the event of a miss, memory access time and cache access time **do overlap.**
- Do not round

8(A) (2 points) What would the miss rate need to be for the effective access time to be 5ns?

Not saved Submit Quiz