# ACTUAL Midterm Exam

This is a preview of the published version of the quiz

Started: Mar 10 at 12:26am

# 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.

| | **8-bit 2s complement signed** | **8-bit unsigned** |
|---|---|---|
| Binary representation of 7 decimal | | ---------- |
| Binary representation of decimal -128 | | ---------- |
| Binary representation of decimal 128 | ---------- | |
| Binary representation of Tmax (maximum signed) | | ---------- |
| Integer (Decimal) value of (-125 - 6) | | ---------- |

**Question 2: Floats**

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

- 12-bit width
- There is s = 1 sign bit
- There are k = 6 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) 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? Answer in decimal.

Question 5 1 pts

**Question 2: Floats**

2(D) (1 points) How many points on the number line represent positive or negative NaNs? Answer in decimal.

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** |
|---|---|---|
| 3/65,536 | | -- |
| +/- Numerator/Denominator<br>Sign:<br>Numerator:<br>/<br>Denominator: | 1011 1101 0000 | -- |

| | | |
|---|---|---|
| Important notes:<br>• Sign should be + or -<br>• Fully reduce the fraction | | |

3. (20 points) **Assembly**

Please consider the following assembly code segment:

```
loop:
        pushq   %r13
        pushq   %r12
        leaq    .LC0(%rip), %r12
        pushq   %rbp
        pushq   %rbx
        subq    $8, %rsp
        call    rand@PLT
        movq    %r12, %rsi
        movl    $1, %edi
        movl    %eax, %ebx
        movl    %eax, %edx
        xorl    %eax, %eax
        call    __printf_chk@PLT
        cmpl    $539959071, %ebx
        jle     .L10
        movl    $5, %ebx
        leaq    .LC1(%rip), %rbp
.L8:
        movq    %rbp, %rdi
        call    puts@PLT
        movq    stdout(%rip), %rdi
        call    fflush@PLT
        subl    $1, %ebx
        jne     .L8
.L9:
        leaq    .LC3(%rip), %rdi
        call    puts@PLT
        movq    stdout(%rip), %rdi
        addq    $8, %rsp
        popq    %rbx
        popq    %rbp
        popq    %r12
        popq    %r13
        jmp     fflush@PLT
```

```
.L10:
        movl    $10, %ebp
        leaq    .LC2(%rip), %r13
.L7:
        movq    %r13, %rdi
        call    puts@PLT
        movq    stdout(%rip), %rdi
        call    fflush@PLT
        call    rand@PLT
        movq    %r12, %rsi
        movl    $1, %edi
        movl    %eax, %ebx
        movl    %eax, %edx
        xorl    %eax, %eax
        call    __printf_chk@PLT
        cmpl    $9999, %ebx
        jle     .L9
        subl    $1, %ebp
        jne     .L7
        jmp     .L9
```

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(10,5)", how many 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".

Question 12 4 pts

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

Consider the following struct:

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

Question 13 4 pts

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

Question 14 4 pts

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

Question 15 4 pts

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

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)**

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** |
| --- |
| short numbersA[  2 ][ 4 ][ 6 ]; // shorts are 2 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. *Hint*: Rows are contiguous in memory.

[ ] Bytes

*Hint:* Think sizeof()

Question 18 2 pts

**Array Arithmetic**

**5(c) (2 points):** Consider the following definitions as implemented on a shark machine, i.e. x86-64. Longs are 8-bytes wide. What is the difference, i.e. number of bytes, between the address of numbers[ 1 ][ 2 ] and the address of numbers[ 2 ][ 1 ]?

```
long numbers[ 4 ][ 5 ];
```

**6. Switch Statement (10 points)**

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

```
Dump of assembler code for function foo:
=> 0x00005555555551d0 <+0>:     endbr64
   0x00005555555551d4 <+4>:     cmp    $0x7,%esi
   0x00005555555551d7 <+7>:     ja     0x555555555203 <foo+51>
   0x00005555555551d9 <+9>:     lea    0xe24(%rip),%rdx   #0x555555556004
   0x00005555555551e0 <+16>:    mov    %esi,%eax
   0x00005555555551e2 <+18>:    movslq (%rdx,%rax,4),%rax
   0x00005555555551e6 <+22>:    add    %rdx,%rax
   0x00005555555551e9 <+25>:    notrack jmp *%rax
   0x00005555555551ec <+28>:    nopl   0x0(%rax)
   0x00005555555551f0 <+32>:    shl    $0x2,%edi
   0x00005555555551f3 <+35>:    lea    0x2(%rdi),%eax
   0x00005555555551f6 <+38>:    ret
   0x00005555555551f7 <+39>:    nopw   0x0(%rax,%rax,1)
   0x0000555555555200 <+48>:    sub    $0x6,%edi
   0x0000555555555203 <+51>:    lea    (%rdi,%rsi,1),%eax    # -3585
   0x0000555555555206 <+54>:    ret
   0x0000555555555207 <+55>:    nopw   0x0(%rax,%rax,1)
   0x0000555555555210 <+64>:    movslq %edi,%rsi
```

```
   0x0000555555555213 <+67>:    sar    $0x1f,%edi
   0x0000555555555216 <+70>:    imul   $0x55555556,%rsi,%rsi
   0x000055555555521d <+77>:    shr    $0x20,%rsi
   0x0000555555555221 <+81>:    mov    %esi,%eax
   0x0000555555555223 <+83>:    sub    %edi,%eax
   0x0000555555555225 <+85>:    ret
   0x0000555555555226 <+86>:    cs nopw 0x0(%rax,%rax,1)
   0x0000555555555230 <+96>:    lea    0xa(%rdi),%eax
   0x0000555555555233 <+99>:    ret
```

And the following memory dump:

```
(gdb) x/16lx 0x555555556004
0x555555556004: 0xfffff1ff      0xfffff22c      0xfffff1ec      0xfffff1ef
0x555555556014: 0xfffff20c      0xfffff1ff      0xfffff20c      0xfffff1fc
0x555555556024: 0x000a6425      0x3b031b01      0x0000003c      0x00000006
0x555555556034: 0xffffeff8      0x00000070      0xfffff028      0x00000098

(gdb) x/16ld 0x555555556004
0x555555556004: -3585   -3540   -3604   -3601
0x555555556014: -3572   -3585   -3572   -3588
0x555555556024: 680997  990059265       60      6
0x555555556034: -4104   112     -4056   152
```

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 integer input values are managed by non-default cases of the switch statement?

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: 32 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 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 25 1 pts

**Part 6(A)(1): Caching**

Given a cache described as follows:

- Number of sets: 8
- Total size: 4,096 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 12-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 model described as follows:

- Number of sets: 8
- Total size: 4,096 bytes (not counting meta data)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 12-bit addresses

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

Question 27 1 pts

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

Given a cache described as follows:

- Number of sets: 8
- Total size: 4,096 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 12-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)(4-9) Caching (12 points, 1 point each)**:

Given a cache described as follows:

- Number of sets: 8
- Total size: 4,096 bytes (not counting meta data like the valid bit or the r, w, and x bits, etc.)
- 2-way set associative
- Replacement policy: Set-wise LRU
- 16-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?**<br>**Select one (per row):** | **Miss Type (Choose N/A for Hit)?**<br>**Select one (per row)** |
|---|---|---|---|
| ---------- | 0x8888 | ---------- | ---------- |
| ---------- | 0x90AA | ---------- | ---------- |
| **7(A)** | 0x88BB | [ Select ] | [ Select ] |
| **7(B)** | 0x1DAA | [ Select ] | [ Select ] |
| **7(C)** | 0x25BB | [ Select ] | [ Select ] |
| **7(D)** | 0x2DAA | [ Select ] | [ Select ] |
| **7(E)** | 0x90DD | [ Select ] | [ Select ] |

| | | | |
|---|---|---|---|
| **7(F)** | 0x1DBB | [ Select ] | [ Select ] |
| **7(G)** | 0x25BB | [ 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 2ns access time.
- The main memory has an access time of 9ns.
- The cache miss rate is 20%.
- In the event of a miss, memory access time and cache access time do **not** overlap.
- Do not round

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

Quiz saved at 12:27am Submit Quiz