CSE 351

Final Exam Review

Final Exam Review

- The final exam will be comprehensive, but more heavily weighted towards material after the midterm

- We will do a few problems from previous years' finals together as a class
  - PLEASE ask questions if you get lost!

Quiz

- We have another quiz we want to spend a few minutes on

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

True

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

True

3. Shifting an int by 3 bits to the left (<< 3) is the same as multiplying it by 8.

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

True

3. Shifting an int by 3 bits to the left (<< 3) is the same as multiplying it by 8.

True

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

True

3. Shifting an int by 3 bits to the left (<< 3) is the same as multiplying it by 8.

True

4. In C, endianness makes a difference in how character strings (char*) are stored.

Quiz

1. A 4-byte integer can be moved into a 32-bit register using a movw instruction.

False

2. On a 64-bit architecture, casting a C integer to a double does not lose precision.

True

3. Shifting an int by 3 bits to the left (<< 3) is the same as multiplying it by 8.

True

4. In C, endianness makes a difference in how character strings (char*) are stored.

False

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

True

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

True

7. An instruction cache takes advantage of only spatial locality.

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

True

7. An instruction cache takes advantage of only spatial locality.

False

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

True

7. An instruction cache takes advantage of only spatial locality.

False

8. Caches are part of the instruction set architecture (ISA) of a computer.

Quiz (cont)

5. In C, storing multi-dimensional arrays in row major order makes it possible for pointer arithmetic to determine the address of an array element.

True

6. A struct can't have internal fragmentation if the elements of the struct are ordered from largest to smallest.

True

7. An instruction cache takes advantage of only spatial locality.

False

8. Caches are part of the instruction set architecture (ISA) of a computer.

False

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

False

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

False

10. On a 64-bit architecture, if a cache block is 32 bytes, and there are 256 sets in the cache, the tag will be 53 bits.

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

False

10. On a 64-bit architecture, if a cache block is 32 bytes, and there are 256 sets in the cache, the tag will be 53 bits.

False

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

False

10. On a 64-bit architecture, if a cache block is 32 bytes, and there are 256 sets in the cache, the tag will be 53 bits.

False

11. A process's instructions are typically in a read-only segment of memory.

Quiz (cont)

9. Caches make computers slower by getting between the CPU and memory.

False

10. On a 64-bit architecture, if a cache block is 32 bytes, and there are 256 sets in the cache, the tag will be 53 bits.

False

11. A process's instructions are typically in a read-only segment of memory.

True

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

True

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

True

14. Two running instances of the same process share the same memory address space.

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

True

14. Two running instances of the same process share the same memory address space.

False

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

True

14. Two running instances of the same process share the same memory address space.

False

15. Java generally has better performance than C.

Quiz (cont)

12. A shared library can be accessed from multiple virtual address spaces, but with only one copy in physical memory.

True

13. Virtual memory allows programs to act as if there is more physical memory than there actually exists on the computer.

True

14. Two running instances of the same process share the same memory address space.

False

15. Java generally has better performance than C.

False

Processes

- List the two important illusions that the process abstraction provides to programs.
- For each illusion, list a mechanism involved in its implementation.

Processes

- List the two important illusions that the process abstraction provides to programs.
- For each illusion, list a mechanism involved in its implementation.

- *1. Logical control flow: the process executes as if it has complete control over the CPU. The OS implements this by interleaving execution of different processes via context-switching(exceptional control flow...).*
- *2. Private linear address space: the process executes as if it has access to a private contiguous memory the size of the virtual address space.*

Virtual Memory

- One purpose of virtual memory is to allow programs to use more memory than is available in the physical memory by storing some parts on disk transparently. Name some *other* useful thing that can be done with the virtual memory system.

Virtual Memory

- One purpose of virtual memory is to allow programs to use more memory than is available in the physical memory, by storing some parts on disk transparently. Name some *other* useful things that can be done with the virtual memory system.

- *1. Sharing of a single physical page in multiple virtual address spaces (e.g., shared library code).*
- *2. Memory protection mechanisms (e.g., page-granular read/write/execute permissions or protecting one process's memory from another).*

TLBs

- Does a TLB (Translation Lookaside Buffer) miss always lead to a page fault? Why or why not?

TLBs

- Does a TLB (Translation Lookaside Buffer) miss always lead to a page fault? Why or why not?

- *No. The TLB caches page table entries. After a TLB miss, we do an in-memory page table lookup. A page fault occurs if the page table entry is invalid.*

Java vs C

- Name some differences between Java references and C pointers.

Java vs C

- Name some differences between Java references and C pointers.

- *1. C allows pointer arithmetic; Java does not.*
- *2. C pointers may point anywhere (including the middles of memory objects); Java references point only to the start of objects.*
- *3. C pointers may be cast arbitrarily (even to non-pointer types); casts of Java references are checked to make sure they are type-safe.*

Structs

Consider the following definition of the struct below, answer the questions regarding the struct.

```
typedef struct data_struct
{
  int a;
  char b[3];
  short c;
  void * d;
} data_struct;
```

1) Assume you have an data_struct array of size four, on the stack diagram on the following page please shade in and label the memory blocks for each field of each struct.

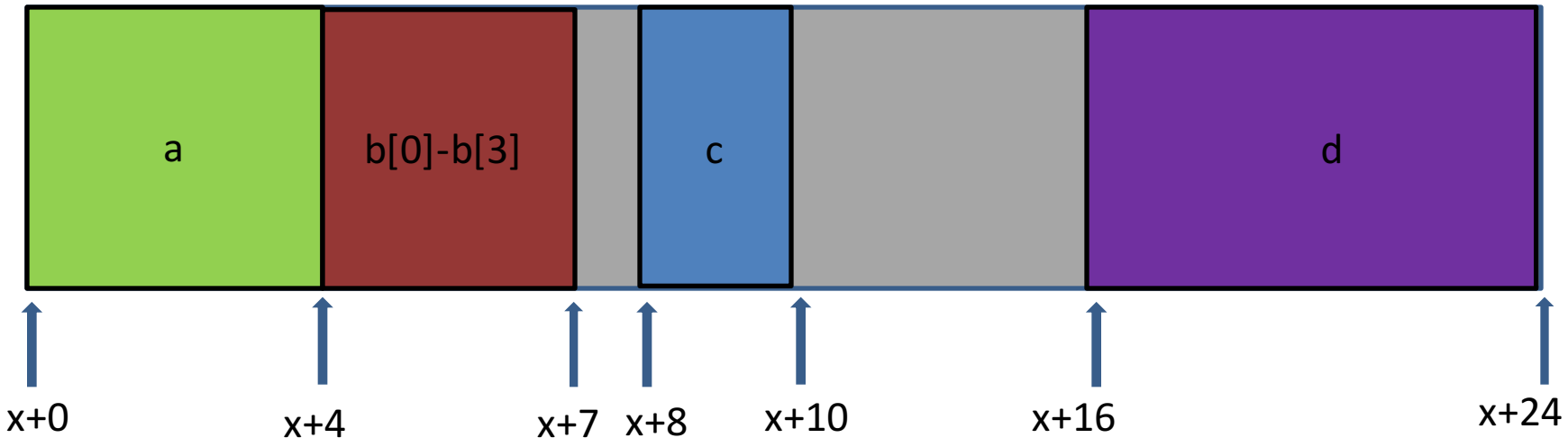
2) What is the total size of this struct?

3) Would re-ordering the fields from largest to smallest reduce the size of the struct?

4) What would be the assembly code for getting the value of field d out of the struct? Assume that the register %rdi points to the beginning of the struct. Return the value in register %rax.

Structs

```
typedef struct data_struct {
 int a;
 char b[3];
 short c;
 void *d;
} data_struct;
```



Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→								
0x08								
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a							
0x08								
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]			
0x08								
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]	b[1]		
0x08								
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]	b[1]	b[2]	
0x08								
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]	b[1]	b[2]	
0x08	c							
0x10								
0x18 (array index 2) →								
0x20								
0x28								

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]	b[1]	b[2]	
0x08	c							
0x10	d							
0x18 (array index 2) →								
0x20								
0x28								

Repeat the process…

Memory Address	+0	+1	+2	+3	+4	+5	+6	+7
0x00 (array index 1)→	a				b[0]	b[1]	b[2]	
0x08	c							
0x10	d							
0x18 (array index 2) →	a				b[0]	b[1]	b[2]	
0x20	c							
0x28	d							

Structs

What is the total size of this struct?

24 bytes, (17 bytes plus 7 byte of internal padding)

Would re-ordering the fields from largest to smallest reduce the size of the struct?

No external fragmentation would still keep the size the same (however remember that you should always order from largest to smallest because it helps more often than not)

What would be the assembly code for getting the value of field d out of the struct? Assume that the register %rdi points to the beginning of the struct. Return the value in register %rax.

```
movq 0x10(%rdi), %rax
ret
```

Structs

```
typedef struct data_struct {
 void * a;
 int b;
 short c;
 char d[3];
} reordered_data_struct;
```