CS61C Summer 2013 Midterm Review Session

Albert, Jeffrey, Justin, Kevin, Sagar, Shaun

Reminders

- Time/Location: **Friday 7/19, 9am-12pm in 1 Pimentel**
- Rules:
    - **Bring**: Double-sided, 8.5"x11" sheet of handwritten notes, pencil, eraser
    - **Provided**: MIPS Green Sheet, Exam with empty space for work, some scratch paper, time-keeping
    - ***DO NOT* Bring**: Books, Printed Notes, Calculators

- Additional resources available on Piazza @366

Outline

1. Warmup (Number Rep, Endianness)
2. C Programming
3. C to MIPS
4. AMAT
5. Floating Point
6. Caches

Warmup: Number Rep

Given the indicated number rep, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=):

a) Unsigned 32-bit numbers

A. 0111 0000 1111 0101 0111 0000 1111 0101

B. 0111 0000 1011 0101 0111 0000 1111 0101

A ____ B

Warmup: Number Rep

Given the indicated number rep, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=):

b) One's complement 32-bit numbers

A. 0000 0000 0000 0000 0000 0000 0000 0000

B. 1111 1111 1111 1111 1111 1111 1111 1111

A ____ B

Warmup: Number Rep

Given the indicated number rep, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=):

c) Two's complement 32-bit numbers

A. 1111 0000 1111 0101 0111 0000 1111 0101

B. 1111 0000 1011 0101 0111 0000 1111 0101

A ____ B

Warmup: Number Rep

Given the indicated number rep, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=):

d) IEEE 754 Single-Precision FP

A. 1111 1111 0111 0101 0111 0000 1111 0101

B. 1111 1110 0111 0101 0111 0000 1111 0101

A ____ B

Warmup: Number Rep

Given the indicated number rep, fill in the blank with exactly one of LESS THAN (<), GREATER THAN (<), or EQUAL TO (=):

e) Another IEEE 754 Single-Precision FP

A. 1000 0000 1111 0101 0111 0000 1111 0101

B. 1000 0001 0111 0101 0111 0000 1111 0101

A ____ B

Warmup: C, Pointers, Endianness

What does C print? Assume that numbers are stored in ***LITTLE-ENDIAN*** format. Assume that ints are 32-bits and chars are 8-bits.

```
int main(int argc, char** argv) {
   unsigned int data[] = {0x01234567,
                          0x89ABCDEF};
   unsigned char* charFront = (char *) data;
   unsigned char* charMid = (char *) &(data
[1]);
   unsigned char* charEnd = (char *)
                                  &(*(data+2));
   // %x means print the variable in hex.
   printf("The first print: %x\n", (int)
                               *(charFront+1));
}
```

Warmup: C, Pointers, Endianness

What does C print? Assume that numbers are stored in ***LITTLE-ENDIAN*** format. Assume that ints are 32-bits and chars are 8-bits.

```
int main(int argc, char** argv) {
   unsigned int data[] = {0x01234567,
                          0x89ABCDEF};
   unsigned char* charFront = (char *) data;
   unsigned char* charMid = (char *) &(data
[1]);
   unsigned char* charEnd = (char *)
                                  &(*(data+2));
   // %x means print the variable in hex.
   printf("The second print: %x\n", (int)
                               *(charFront+4));
}
```

Warmup: C, Pointers, Endianness

What does C print? Assume that numbers are stored in ***LITTLE-ENDIAN*** format. Assume that ints are 32-bits and chars are 8-bits.

```
int main(int argc, char** argv) {
   unsigned int data[] = {0x01234567,
                          0x89ABCDEF};
   unsigned char* charFront = (char *) data;
   unsigned char* charMid = (char *) &(data
[1]);
   unsigned char* charEnd = (char *)
                                  &(*(data+2));
   // %x means print the variable in hex.
   printf("The third print: %x\n", (int)
                             charMid[1]);
}
```

Warmup: C, Pointers, Endianness

What does C print? Assume that numbers are stored in ***LITTLE-ENDIAN*** format. Assume that ints are 32-bits and chars are 8-bits.

```
int main(int argc, char** argv) {
   unsigned int data[] = {0x01234567,
                          0x89ABCDEF};
   unsigned char* charFront = (char *) data;
   unsigned char* charMid = (char *) &(data
[1]);
   unsigned char* charEnd = (char *)
                                  &(*(data+2));
   // %x means print the variable in hex.
   printf("The fourth print: %x\n", (int)
                                *(charMid-4));
}
```

Warmup: C, Pointers, Endianness

What does C print? Assume that numbers are stored in ***LITTLE-ENDIAN*** format. Assume that ints are 32-bits and chars are 8-bits.

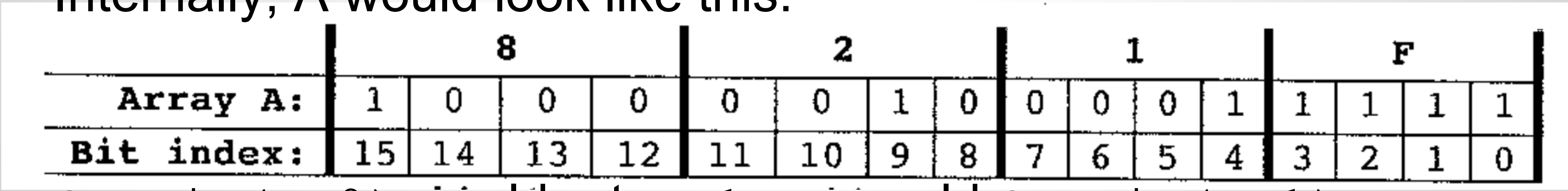
```
int main(int argc, char** argv) {
   unsigned int data[] = {0x01234567,
                          0x89ABCDEF};
   unsigned char* charFront = (char *) data;
   unsigned char* charMid = (char *) &(data
[1]);
   unsigned char* charEnd = (char *)
                                  &(*(data+2));
   // %x means print the variable in hex.
   printf("The fifth print: %x\n", (int)
                               charEnd[-1]);
}
```

C Programming

We wish to implement a bit array, where we can read and write a particular bit. Normally for read/write array accesses, we would just use bracket notation (e.g., `x=A[5]`; `A[5]=y`;), but since a bit is smaller than the smallest datatype in C, we have to design our own `GetBit()` and `SetBit()` functions. We'll use the following typedefs to make our job easier:

```
typedef uint8_t bit_t; // If a single bit,
value is in the least significant bit
typedef uint32_t index_t; // The index into a
bit_t array to select which bit is used
```

E.g. imagine a 16-bit array: bit_t A[2]; A[1]=0x82; A[0]=0x1F; Internally, A would look like this:



	8				2				1				F			
Array A:	1	0	0	0	0	0	1	0	0	0	0	1	1	1	1	1
Bit index:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

`GetBit(A,0)` would return `1`, as would `GetBit(A,1)`

`GetBit(A,5)` would return `0`, as would `GetBit(A,6)`

C Programming

Questions:

1. How much space would the largest usable bit array take up? "Usable" means that we could read and write every bit in the array. Express your answer in IEC format. E.g., 128KiB, 32TiB, etc.
2. Write `SetBit` in C. You may not need to use all the lines.

```
void SetBit(bit_t A[], index_t n, bit_t b) {
    //YOUR CODE HERE


}
```

C Programming

Questions:

3. Write `GetBit(bit_t A[], index_t n)` in <u>MAL</u>; `$v0` should be `1` if the bit is on, and `0` if it's off. Hint: It might be helpful if you start from the `srlv` and work backwards.

```
GetBit: _____ $t0, ____________
        _____ $t1, ____________
        _____ $t2, ____________
        _____ $t3, ____________
        srlv  $v0, $t2, $t3 #srlv: rd=rt >> rs

        _____ _________________
        jr $ra      #$v0 should be either 0 or
1
```

C to MIPS

Given below is a C code fragment from a binary search tree routine. It returns the array index of an integer that matches the given search element, and for simplicity, assume the element is always present. Convert this code to MIPS, using as few instructions as possible:

```
int helperPBST (int *arr, int pos, int elem) {
   int temp = pos - 1;
   if (elem == *(arr + temp)) {
      return pos;
   } else if (elem > *(arr + temp)) {
      return helperPBST(arr, pos*2+1, elem);
   } else {
      return helperPBST(arr, pos*2, elem);
   }
}
```

C to MIPS

Fill in the MIPS:

```
# $a0 is arr, $a1 is pos
# $a2 is elem
pbst:
   addi $t0, $a1, -1
   sll  ______________
   add  ______________
   lw   ______________
   beq $a2, $t1, success
   slt $t3, $a2, $t1
   beq $t3, $0, right

   ____________________

   ____________________
```

```
right:

   __________________

   __________________

   __________________
success:

   __________________
   jr $ra
```

AMAT

Bob's computer specs are currently:

- Unified L1 Cache
- L1 Cache hit rate of 90%
- L1 Cache hit time of 1 cycle
- Miss penalty to main mem of 100 cycles
- Ideal CPI of 1

a) What is AMAT?

b) If he runs a program that has 50% loads/stores, what is his program's CPI?

AMAT (continued)

Bob's computer specs are currently:

- Unified L1 Cache
- L1 Cache hit rate of 90%
- L1 Cache hit time of 1 cycle
- Miss penalty to main mem of 100 cycles
- Ideal CPI of 1

Disgusted at his slow computer, he requests you improve it by adding an L2 cache. He wants you to cut down his AMAT to 6.

c) The L2 cache you have in mind has a Local Miss Rate of 35%. What is the worst hit time it can have while still meeting Bob's request?

AMAT (continued)

Bob's computer specs are currently:

- Unified L1 Cache
- L1 Cache hit rate of 90%
- L1 Cache hit time of 1 cycle
- Miss penalty to main mem of 100 cycles
- Ideal CPI of 1

Disgusted at his slow computer, he requests you improve it by adding an L2 cache. He wants you to cut down his AMAT to 6.

d) Bob doesn't like it, so you set Bob up with a different L2 cache with a Hit Time of 10 cycles. Now his system has a global miss rate of 6%. What is his new CPI?

AMAT (continued)

Bob's computer specs are currently:

- Unified L1 Cache
- L1 Cache hit rate of 90%
- L1 Cache hit time of 1 cycle
- Miss penalty to main mem of 100 cycles
- Ideal CPI of 1

Disgusted at his slow computer, he requests you improve it by adding an L2 cache. He wants you to cut down his AMAT to 6.

e) What is the relative performance of his upgraded computer versus his old one? You may leave the value as a ratio.

Floating Point

Consider the following 10-bit FP format. It contains the same fields (S, E, M) and follows the same general rules as 32-bit IEEE 754 (denorms, biased exp., non-numerics). It simply allocates its bits differently.

S	EEE	MMMMMM

Number rep'd by `0x00`: `0`

# Bits in Mantissa: `6`

a) What is the exponent bias?

b) What is the implicit exponent for denorms?

c) How many numbers can we represent in the range [2, 8)?

d) What is the largest x such that x + 0.5 = 0.5?

Caches

a) The AMAT has three components: (hit time, miss rate, and miss penalty). For each of the following cache optimizations, indicate which component of the AMAT equation may be improved.

- Using a second-level cache
- Using larger blocks
- Using a smaller first-level cache
- Using a larger first-level cache

Caches (continued)

b) Given a direct-mapped cache, initially empty, and the following memory access pattern (all byte addresses and 32-bit word accesses, 32-bit addresses)

8, 0, 4, 32, 36, 8, 0, 4, 16, 0

What is the hit rate, miss rate, and what blocks are in the cache after these accesses if:

i) The cache has 8 32-bit blocks.

ii) The cache has 4 32-bit blocks.

Caches (continued)

c) Consider a write-allocate, write-back, direct-mapped cache with 16 byte blocks and 64*2^10 bytes of data bits. Assume a byte-addressed machine with 32 bit addresses.

i) Draw the partitioned address and label each field
ii) What are the TIO values for the addr `0xDEADBEEF`?
iii) How many cache management bits are there for each block? List them.
iv) What is the total number of bits (data AND cache management) that comprise the cache?

That's it for this session!

We hope you found our pointers (haha) useful and good luck on your midterm!

Acknowledgements

- Warmup - Fa12 MT
- C Programming - ???
- C to MIPS - Fa12 MT
- AMAT - Su11 MT
- Floating Point - ???
- Caches - Sp11 MT1