Computer Systems Organization
CSCI-UA.0201-001 Spring 2022
Final Exam ANSWERS

Write your answers on this exam paper. You can use the back of the sheets for scratch.

1. (10 points) True/False. Please circle the correct response.

   a. T A write-through cache doesn't need a dirty bit associated with each cache entry.

   b. F The output of a clocked latch (aka "D-latch") changes only when the clock goes from high voltage to low voltage.

   c. F The instruction "leaq 16(%rcx),%rdx" on the x64 processor could cause a cache miss if the contents of the memory location at address %rcx+16 is not in the cache.

   d. T Assuming a two's complement representation of signed integers, shifting a signed integer variable left by one bit in C could change the sign of the value of the variable.

   e. F Assuming a two's complement representation of signed integers, shifting a signed integer variable right by one bit in C could change the sign of the value of the variable.

   f. T The instruction "addq -8(%rbp),%rax" will cause the write_enable line for the register file to be 1.

   g. F A callee-saved register must be saved in a function even if the function is only reading the register (not writing it).

   h. T Local variables in a function that are allocated on the stack will be at negative offsets from %rbp on the x64 processor.

   i. F Formal parameters to a function that are passed on the stack (after the ones passed in registers) will be at negative offsets from %rbp on the x64 processor.

   j. F In C, the operation "x & 0x8" will evaluate to 0 if bit 3 (the fourth lowest bit) of x is 0 and will evaluate to 1 if bit 3 of x is 1.

2. (10 points: a is 4 points, the rest 2 points each) Fill in the blanks on this sheet.

   a. Write the number E32A hex in binary: 1110 0011 0010 1010.

   b. log 16T = 44.

   c. In order to access all bytes in a 64GB memory, an address must have at least 36 bits.

   d. If a signed integer variable (using two's complement) is represented by 42 bits, how many different negative numbers could that variable take on? Answer in multiples of K, M, G, or T. $2^{42}/2 = 2^{41} = 2T$.

3. (10 points) Write a C function whose prototype is

```
int intlog(unsigned int num);
```

and which returns the largest integer less than or equal to the log (base 2) of `num`. For example, `intlog(72)` should return 6, since $6 \leq \log 72 < 7$.

Answer:

```
int intlog(unsigned int num) {
  int log = -1;  // not really valid, assuming num > 0
  while (num != 0) {
    num >>= 1;
    log++;
  }
  return log;
}
```

4. (15 points: a is 5 points, b is 10 points) Given the following definition of a `QUEUE` struct type used for a queue,

```
typedef struct {
   long head;
   long tail;
   char *buff[1000];
} QUEUE;
```

where `buff` is an array of strings, `head` is the index of the first string in `buff`, and `tail` is the index of the last string in `buff`,

a. Write a C function whose prototype is

```
void insert_string(char *str, QUEUE *q);
```

that takes a string (char pointer) and a pointer to a queue and inserts the string into the queue's `buff` array after the last string already in the queue. Be sure to update the fields of the queue as necessary. You do not need to check if there is space available in `buff`, you can just assume there is. This is a very short function.

Answer:

```
void insert_string(char *str, QUEUE *q)
{
  q->tail++;
  q->buff[q->tail] = str;
}
```

b. Translate your C function into x86-64 assembly code for either Unix (e.g. MacOS or Linux) or Cygwin/Windows. An assembly reference sheet and the calling conventions for Unix and Windows are provided at the end of this exam. You can assume that the fields of a `QUEUE` struct are contiguous (no space between them).

Answer:

```
_insert_string:

     pushq %rbp
     movq  %rsp, %rbp

     # str is in %rdi
     # q is in %rsi
     # tail is at offset 8
     # buff is at offset 16

     incq  8(%rsi)          # q->tail++;
     movq  8(%rsi),%rcx     # %rcx holds q->tail
     movq  %rdi,16(%rsi,%rcx,8)  # q->buff[q->tail] = str;
                                 # each element is 8 bytes.
     popq  %rbp
     retq
```

5. (15 points: 5 points each question)

a. These questions relate to direct-mapped caches:

i. In a direct mapped cache whose capacity is K bytes (where K is a power of two), with one word per cache line, what is the relationship between the addresses of two words in memory that map to the same entry in the cache?

Answer: The addresses are a multiple of K apart. That's because the cache index is computed as address mod size_of_cache.

ii. Suppose an address is 32 bits and there is a 1MB direct-mapped cache (where a word is 4 bytes). Not knowing anything else about the cache (e.g. size of the cache line), can you determine how many bits of an address is needed for the tag? If not, explain why not. If so, give the number of tag bits.

Answer:
As indicated in the lecture, the tag is 30-log N bits, where N is the number of words in the cache, so in this case, the tag is $30\text{-}\log(2^{20}/4) = 30 - \log(2^{18}) = 30 - 18 = 12$.

Another way to look at it is that with a 1MB cache, to select a particular byte from the cache requires $\log(1M) = \log(2^{20}) = 20$ bits, no matter how those bits are divided up into byte offset and word offset. The remaining 12 bits in the address must be the tag bits.

iii. Given two caches that are identical in total capacity (number of bytes of data) and in the size of a cache line, where one is a direct-mapped cache and the other is a 4-way set associative cache, what is the relationship between the number of bits of an address used for the cache index for the direct mapped cache and the number of bits used for the set index in the set-associative cache? Be as specific as possible.

Answer: Both caches have the same number of cache entries. Since, in the 4-way set-associative cache, there are 4 cache entries per set, there are ¼ of the number of sets than there are cache entries. Therefore, since log (n/4) = log n – 2, there are two fewer set index bits in the set-associative cache than there are cache index bits in the direct-mapped cache.

b. Suppose that, on a computer with only one cache, the following holds:

- the cache has a 90% hit rate for <u>instructions</u>,
- the cache has an 80% hit rate for <u>data</u>
- if an instruction does not cause a cache miss, executing the instruction takes one cycle (there is no penalty),
- the cache miss penalty is 200 cycles

What is the overall execution time (in cycles) of a program that executes I instructions, where 25% of the instructions access data in memory? Show your work.

Answer:

$$\begin{aligned}\text{Total Cycles} &= I + (I * .1 * 200) + (I * .25 * .2 * 200)\\ &= I + (I * 20) + (I * 10)\\ &= 31\, I\end{aligned}$$

c. On a machine with 32-bit words and a 4MB 4-way set-associative cache with 8 words per cache line, indicate how the bits of an address is divided up into a byte offset, word offset, set index, and tag, indicating the number of bits and the bit position of each. Show your work.

Answer:
There are 4 bytes per word, so the byte offset is two bits at bits 0-1.

There are 8 words per cache line, so the word offset is 3 bits at bits 2-4.

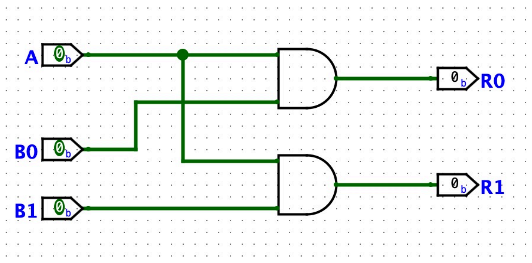
The cache size is 4MB = 1M words = 2^{20} words. Since there are 8 words per cache line, there are $2^{20}/2^3 = 2^{17}$ cache lines. Since a 4-way set-associative cache has 4 cache lines per set, there are $2^{17}/2^2 = 2^{15}$ sets. Thus, log 2^{15} = 15 bits are needed for the set index, at bits 5-19.

The remaining 12 bits are used for the tag, at bits 20-31.

6. (10 points: 5 points each) Digital Logic

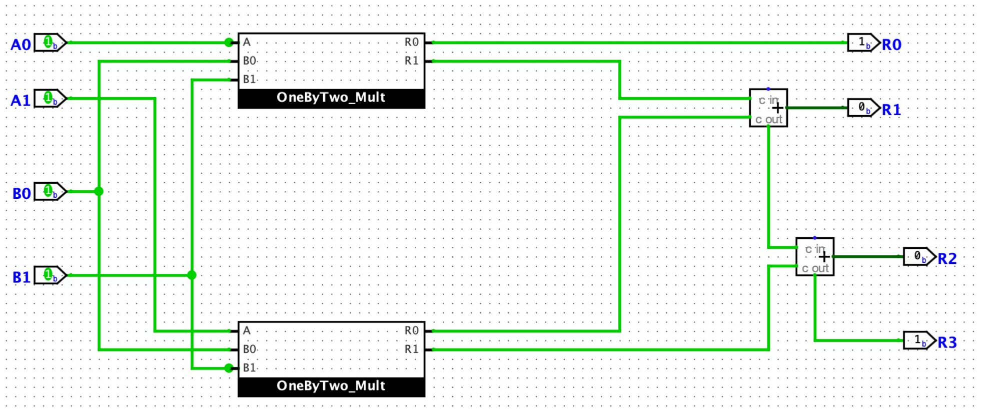
a. Draw a circuit (consisting only of AND, OR, and/or NOT gates) that multiplies a 1-bit input A by a 2-bit input B (consisting of bits B1 and B0), resulting in two bits of output. Note: it's a very simple circuit!

Answer:



b. Using your 1bit x 2bit multiplier, above, build a circuit that multiplies two 2-bit inputs to give a 4-bit output. You can also use your 1-bit adder from Homework 2 (taking a carry_in and two 1-bit inputs and producing a 1-bit result and a carry_out). You don't have to build the adder and can draw the adder and the 1bit x 2bit multiplier as boxes (and you can use as many of these as you like). Hint: Write out a simple two-bit binary multiplication example and see what operations are performed in order to do this.

Answer:



7. (10 points: a is 2 points, b is 8 points) Processor Datapath

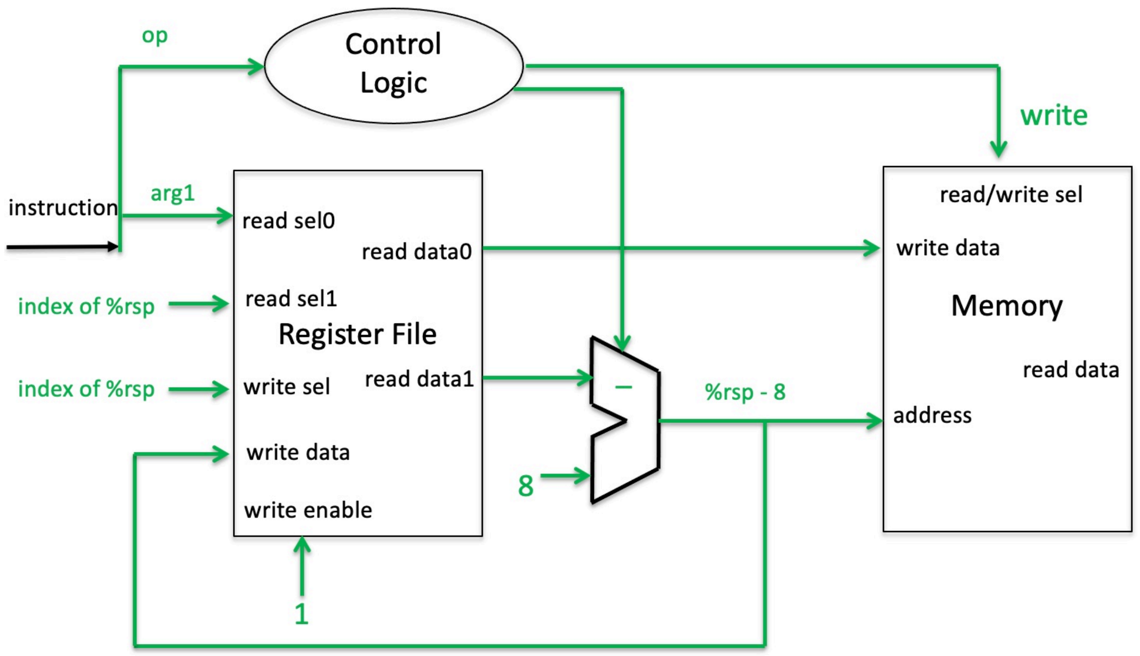
a. Consider the follow assembly instruction: `pushq %rax`

i. What affect does the `pushq` instruction have on memory and any register?

Answer: It decrements %rsp by 8 and writes the specified register (%rax in this case) to (%rsp), the location in memory at the address specified by %rsp.

ii. Below is the skeleton of the datapath drawn in class for moving data between a register and a memory location (e.g "movq %rax,8(%rcx)" and "movq 8(%rcx),%rax"). Assuming that all registers (other than %rip) are in the register file, fill in the diagram below with wires would be needed to implement the above `pushq` operation. Be sure to label each wire with the value (or a description of the value) that the wire would need to carry. Assume that the instruction contains only <u>one</u> operand, which specifies the register (e.g. %rax, above) whose value is being pushed. If the ALU is used, specify the operation it performs. For simplicity, don't worry about the clock.

Answer:



Note: The "index of %rsp" values come from the Control Logic. Those wires are not shown for simplicity.