University of California, Berkeley
College of Engineering
Computer Science Division — EECS

Fall 1999 John Kubiatowicz

Prerequisite Quiz

September 1, 1999
CS152 Computer Architecture and Engineering

This prerequisite quiz will be used in determining class admissions. The use of is not allowed during this quiz. Good Luck!

Your Name:	
SID Number:	
Discussion Section:	

1	
2	
3	
4	
Total	

[This page left for π]

3.14159265358979323846264338327950288419716939937510582097494 4

1) Suppose that we start with the following C program for fibonacci:

```
int fib (int n) {       /* n >= 0 */
   int temp;
   int sum;

   if (n <= 1) {
      return 1;
   } else {
      temp = fib (n-1);
      sum = temp + fib (n-2);
      return sum;
   }
}
```

Here is a translation of the above program into MIPS assembly language.

```
fib:  addi  $v0, $0, 0
      slti  $t0, $a0, 1
      bne   $t0, $0, end

fib1: addi  $sp, $sp, -12
      sw    $a0, 4($sp)
      addi  $a0, $a0, -1
      j     fib

      add   $s0, $v0, $0

      lw    $a0, 4($sp)
      addi  $a0, $a0, -2
      jr    fib1

      add   $v0, $s0, $v0

end:  lw    $ra, 0($sp)
      addi  $sp, $sp, 12
      jr    $ra
```

There are 8 mistakes in this translation. Fix them in the above listing. *Assume that MIPS register conventions must be maintained throughout the execution.*

2) Assume that we have a 16 bit system that uses signed, two's-complement integers. Perform the following conversions:

-13_{10} to base 2:

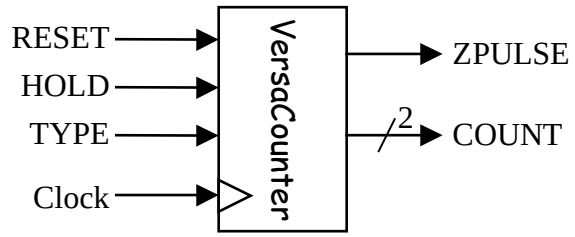
$EFA2_{16}$ to base 8:

34_{10} to base 16:

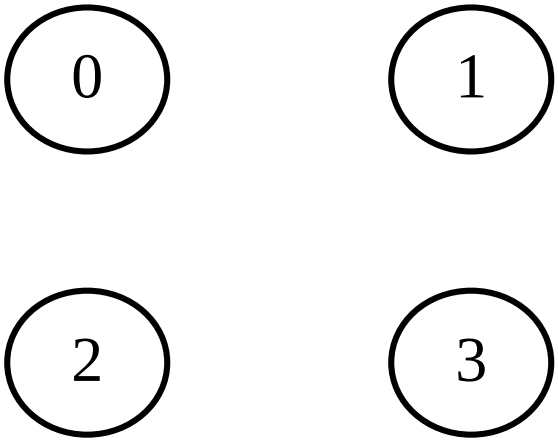
$(1111111100111100)_2$ to base 10:

234_8 to base 10:

3) In this problem, you must design a finite state machine (FSM) that either counts in normal binary (*i.e.* 0, 1, 2, 3, 0, 1, ...) or as a Grey code (0, 1, 3, 2, 0, 1, 3,...) depending on the "TYPE" signal (TYPE=1 ⇒ Grey code). Further, when "HOLD" is asserted, it will stop counting (freeze the count). Whenever the count is "0", it asserts the "ZPULSE" signal for *only one cycle* (independent of HOLD). We will call this the VersaCounter. A diagram for this machine is shown below:



a) Complete the Following State Transition Diagram for the VersaCounter's count output. Ignore the "ZPULSE" signal. Include "HOLD", and "TYPE" as inputs, and ignore RESET for the moment:



b) Construct a State Transition Table for this FSM.

c) Derive Next-State Logic Equations given the state transition table. Include the RESET signal in your equations.

d) Construct a Karnaugh map(s) to minimize the next-state logic equations. Derive the resulting simplified equations.

e) Draw a small circuit that will take a clock and the 2-bit count value as input and will output the "ZPULSE" signal. ZPULSE is high for the very first cycle that the 2-bit count is zero, and deasserted otherwise. *Make this as simple as possible.* You may use falling-edge triggered flip-flops, inverters, and 2, 3, or 4-input NAND gates. Don't forget to include the RESET signal. ZPULSE may be asserted during reset, but should definitely be asserted for the first cycle after reset. (This should **not** be hard to do).

f) Implement the complete VersaCounter using falling-edge triggered flip flops, inverters, and 2, 3, or 4-input NAND gates. Minimize the number of gates that you use. Clearly label input and output signals. It should have as input: CLOCK, HOLD, TYPE, and RESET. It should have as output a 2-bit count value and a 1-bit "ZPULSE" value.

4) Assume that we have a 32-bit processor with 32-bit words, and that this processor is byte-addressed (i.e. addresses specify bytes). Now, suppose that this processor has a 32-word, *direct-mapped* cache with 2-word cache lines.

a) Split the 32-bit address into "tag", "index", and "cache-line offset" pieces. We start you off with one of these.

tag:
index:
cache-line offset: **bits 2 – 0**

b) Assume that the processor makes a series of byte accesses to memory (shown to the left below). Label each memory reference address as a Hit (H) or a Miss (M). Also, show the *final* contents of the cache in the table at the right. Fill in values for the TAG and words of the cache. Use the notation "M[i]" to mean the word in memory at address i (keep these word aligned!). Include no more than one value per word (words can be left blank).

Reference Address	Hit or Miss?
$138_{10}(08A_{16})$:	
$60_{10}(03C_{16})$:	
$56_{10}(038_{16})$:	
$140_{10}(08C_{16})$:	
$419_{10}(1A3_{16})$:	
$484_{10}(1E4_{16})$:	
$142_{10}(08E_{16})$:	
$392_{10}(188_{16})$:	
$175_{10}(0AF_{16})$:	
$140_{10}(08C_{16})$:	
$344_{10}(158_{16})$:	
$487_{10}(1E7_{16})$:	
$396_{10}(18C_{16})$:	
$376_{10}(178_{16})$:	
$380_{10}(17C_{16})$:	
$160_{10}(0A0_{16})$:	

Index	TAG Value	Word 0	Word 1
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
A			
B			
C			
D			
E			
F			

c) Calculate the cache hit rate (you can leave as a fraction).

[This page left for scratch]

[The End]