University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Spring 2015 Instructors: Krste Asanović, Vladimir Stojanovic 2015-05-15

CS61C FINAL

After the exam, indicate on the line above where you fall in the emotion spectrum between "sad" & "smiley"...

Last Name	
First Name	
Student ID Number	
CS61C Login	`cs61c-`
*The name of your **SECTION** TA (please circle)*	David | Donggyu | Fred | Jeffrey | Martin Nolan | Sagar | Shreyas | William
Name of the person to your Left	
Name of the person to your Right	
*All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61C who have not taken it yet. **(please sign)***	

Instructions (Read Me!)

- This booklet contains 14 numbered pages including the cover page. **The back of each page is blank and can be used for scratch-work, but will not be looked at for grading.**
- Please turn off all cell phones, smartwatches, and other mobile devices. Remove all hats & headphones. Place your backpacks, laptops and jackets under your seat.
- You have 180 minutes to complete this exam. The exam is closed book; no computers, phones, or calculators are allowed. You may use three handwritten 8.5"x11" pages (front and back) of notes in addition to the provided green sheet.
- There may be partial credit for incomplete answers; write as much of the solution as you can. We will deduct points if your solution is far more complicated than necessary. When we provide a blank, please fit your answer within the space provided. "IEC format" refers to the mebi, tebi, etc prefixes.
- **You must complete ALL THE QUESTIONS, regardless of your score on the midterm.** Clobbering only works from the Final to the Midterm, not vice versa. You have 3 hours... relax.

	M1-1	M1-2	M1-3	M2-1	M2-2	M2-3	M2-4	F1	F2	F3	Total
Points Possible	9	9	9	10	4	8	12	9	10	10	90
Points Earned											

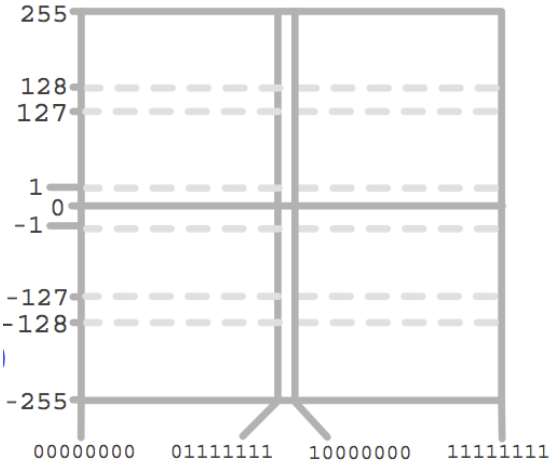
SID: _____

M1-1: *I smell a potpourri section covering midterm one…* (9 points)

a) Which of the following number representations give **`0xFFFFFFFE`** the **most positive** value when converted to decimal?

A) Bias (with standard bias) B) Unsigned C) Two's complement D) Sign and Magnitude

b) Consider a plot that shows the mapping between 8-bit two's complement binary numbers and their decimal equivalents (i.e. binary is on the x-axis and decimal is on the y-axis). Fill in the plot to the left and answer the following questions.



i) Fill in the plot to the left.
ii) Describe (in binary) where discontinuities occur in the plot, if any:

iii) What are the most positive and most negative decimal values that this representation can store?

c) Consider the C code below. Indicate where the values on the right live in memory (using **(S)**tack, **(H)**eap, s**(T)**atic, or **(C)**ode). Assume no registers are used:

```
#define a 10
int b = 0;

int main(int argc, char** argv) {
   int c = a;
   char d[10];
   int* e = malloc(sizeof(int));
}
```

a: _____

b: _____

***d:** _____

***e:** _____

e: _____

d) Convert the following instructions from TAL to hex or vice versa. Use register names when possible.

i) `lw $s0, 0($a0)`

ii) `0x02021021`

SID: _____

M1-2: *I'll believe it when I C it* (9 points)

Your friend wants to take 61C next semester, and is learning C early to get ahead. They try to implement the ROT13 function as practice, but the code they wrote has some bugs, so you've been called in, as the C expert, to help debug their program, reproduced below:

```
1  /* Applies the ROT13 cipher. rot13("happytimes") == "uncclgvzrf" */
2  void rot13(char *str) {
3      while (*str) {
4          if (str >= 'a' && str <= 'z') {
5              *str = (*str + 13) % 26;
6          }
7          str++;
8      }
9  }
10
11 int main(int argc, char *argv[]) {
12     char a[] = "happy";
13     char b[] = "times";
14     char *s = "XXXXXXXXXXXX"; // 12 X's
15
16     // Apply cipher to a and b.
17     rot13(a);
18     rot13(b);
19     printf("%s%s\n", a, b);
20
21     // Concatenate and place in s.
22     int i = 0;
23     for (int j = 0; a[j]; ) s[i++] = a[j++];
24     for (int j = 0; b[j]; ) s[i++] = b[j++];
25
26     printf("%s\n", s);
27 }
```

a) You want to impress your friend, so you predict the result of executing the program as it is written, just by looking at it. If the program is guaranteed to execute without crashing, describe what it prints, otherwise explain the bug that may cause a crash.

b) Now, fix all the errors in the program so that it executes correctly. Fill in the corrections you made in the table below. You may not need all the rows.

Line #	Insert Before / Replace / Delete	Change (Explanation or Code)

SID: _____

M1-3: *I don't want to MIPS a thing* (9 points)

The following C code recursively sums the elements in an array of length `n`.

```
int32_t sum_arr(int32_t *arr, size_t n) {
    if (n) {
        return sum_arr(arr + 1, n - 1) + arr[0];
    }
    return 0;
}
```

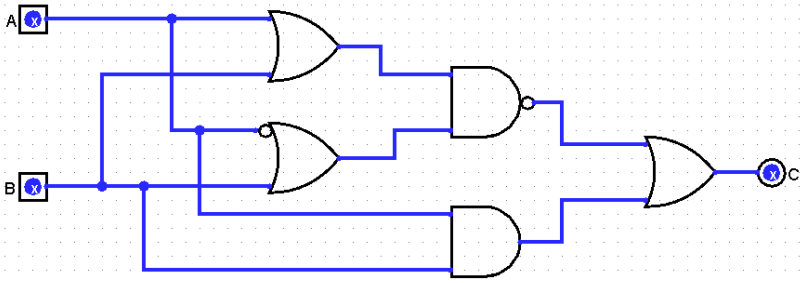
Translate `sum_arr` into MIPS below. Your code must follow all function calling conventions, and you may not use any pseudoinstructions. You may not need every blank.

```
sum_arr:  _____________ non_zero
          addu $v0, $0, $0
          jr $ra
non_zero: _____________________
          _____________________
          _____________________
          _____________________
          addiu $s0, $a0, 0       # Store arr into $s0
          _____________________
          _____________________
          _____________________
          _____________ sum_arr   # Make the recursive call
          _____________________   # Then add arr[0] to the result
          _____________________
          _____________________
          _____________________
          _____________________
          _____________________
          jr $ra
```

SID: _____

M2-1: *I couldn't come up with a clever title for SDS.* (10 points)

a) Give the simplest Boolean expression for the following circuit in terms of A and B, using the minimum number of AND, OR, and NOT gates:



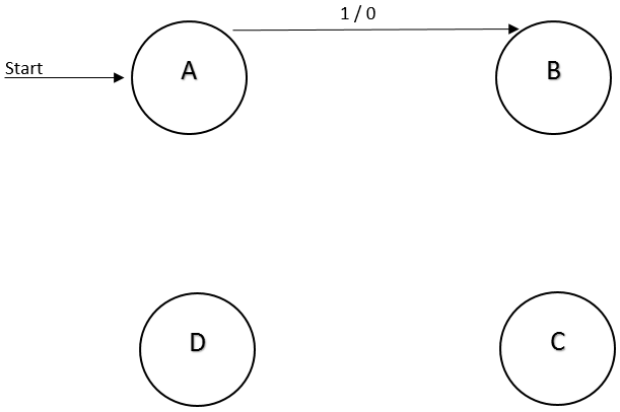
C = _____

(You must show your work above to earn points.)

b) Using as few states as possible, complete the transition table for an FSM that takes an input with 3 values: 0, 1, or 2. The machine will output a 1 when the sum of the inputs seen so far is divisible by 3. Otherwise it should output a 0.

Assume you have seen no digits at the start state. You might not need all of the states, and you should not draw additional states. You must represent your FSM using the table to the left, **the table is the only part that will be graded.** The first transition has been filled in for you.

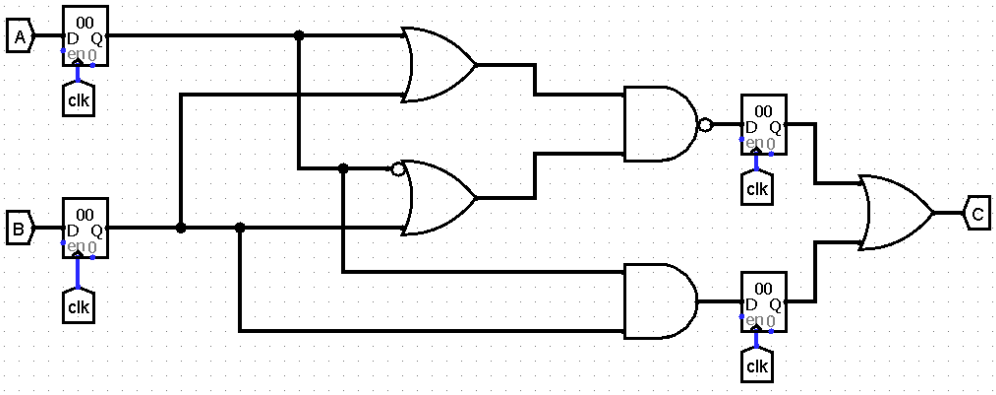
Current State	Input/Output	Next State
A	1/0	B



SID: _____

M2-1: (continued)

c) Suppose we add registers to the unoptimized circuit in part A to increase the clock rate (this modification is shown below). What is the longest clock-to-Q that the registers on inputs A and B can have that will result in correct behavior when the circuit is clocked at 10 MHz?



Assumptions:

- Assume that clock-to-Q > hold time
- All registers have a setup time of 2 ns
- All logic gates have a delay of 25 ns
- Bubbles on gates do not introduce additional delay

Answer: _____

M2-2: Float like a butterfly and sting like an IEEE (4 points)

Let's take another look at the IEEE754 standard for single-precision floating-point numbers. [x, y) represents a range where x is included and y is not.

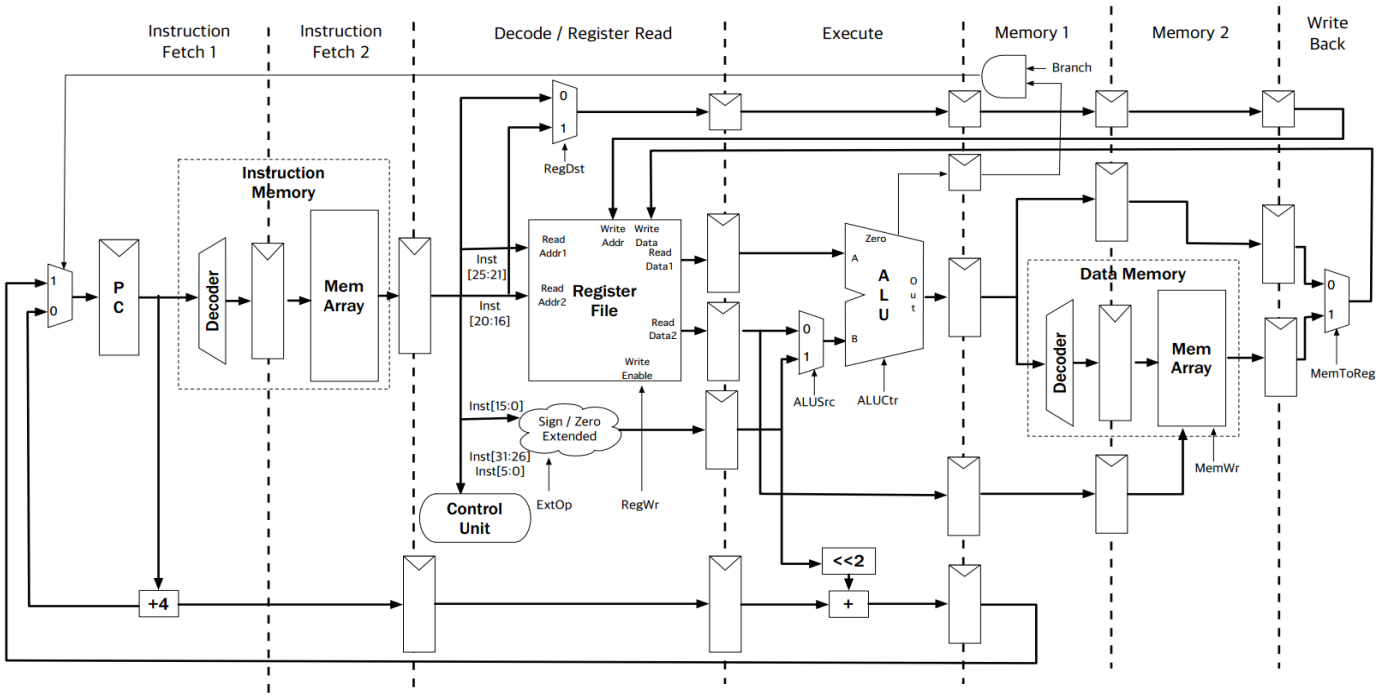
a) How many floats are representable in the interval [0.5, 1)? Answer: ____ · 2^____

b) How many floats are representable in the interval [0, 0.5)? Answer: ____ · 2^____

SID: _____

M2-3: If this exam were a CPU, you'd be halfway through the pipeline (8 points)

We found that the instruction fetch and memory stages are the critical path of our 5-stage pipelined MIPS CPU. Therefore, we changed the IF and MEM stages to take **two** cycles while increasing the clock rate. You can assume that the register file is written at the falling edge of the clock.



Assume that no pipelining optimizations have been made, and that branch comparisons are made by the ALU. Here's how our pipeline looks when executing two add instructions:

Clock Cycle #	1	2	3	4	5	6	7	8
add $t0, $t1, $t2	IF1	IF2	ID	EX	MEM1	MEM2	WB	
add $t3, $t4, $t5		IF1	IF2	ID	EX	MEM1	MEM2	WB

Make sure you take a careful look at the above diagram before answering the following questions:

a. How many stalls would a data hazard between back-to-back instructions require?

b. How many stalls would be needed after a branch instruction?

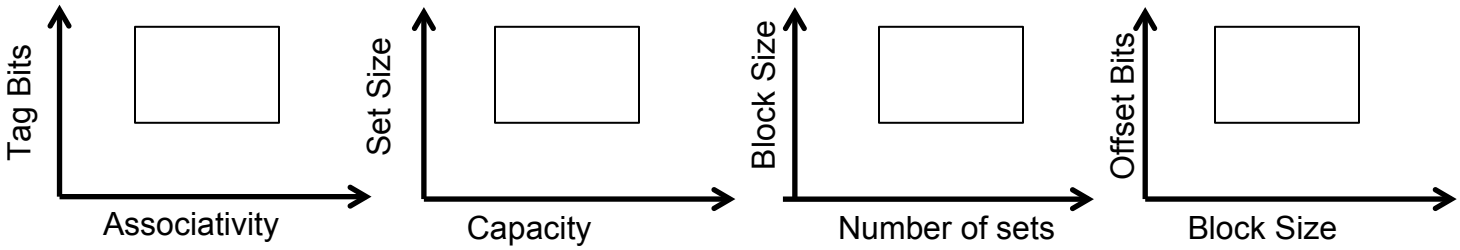
c. Suppose the old clock period was 150 ns and the new clock period is now 100ns. Would our processor have a significant speedup executing a large chunk of code…
   i. Without any pipelining hazards? Explain your answer in 1-2 sentences.

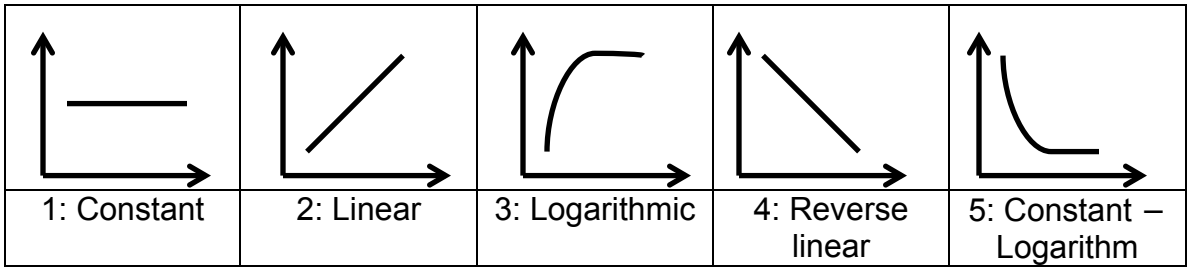
   ii. With 50% of the code containing back-to-back data hazards? Explain your answer in 1-2 sentences.

SID: _____

M2-4: Some say there's nothing better than cold, hard cache (12 points)

a) What shape do the following trade-off curves have? Select a shape and enter its number into the box for each of the graphs. Unless they are the parameters being varied, assume that associativity, capacity and block size are constant. You should assume that the axes are linear.





1: Constant	2: Linear	3: Logarithmic	4: Reverse linear	5: Constant – Logarithm

b) Consider a system with inclusive L1 and L2 caches with 4B cache block size. Assume we have 1 MiB of on-chip memory available and want to determine how much of this memory we should give to the L1 cache and how much to the L2 cache. We will try to minimize the AMAT to do so.

Assume both caches are fully associative with LRU replacement. Their combined capacity is 1MiB (excluding tags and meta-data). **You can consider all miss rates approximate.**

Say you are running the following program starting from cold L1 and L2 caches:

```
#define ARRAY_SIZE 256*1024
int a[ARRAY_SIZE];
int sum = 0; // assume sum, i, and j are stored in registers

for (int i = 0; i < 100000; i++) {
  for (int j = 0; j < ARRAY_SIZE; j++) sum += a[j];
  for (int j = ARRAY_SIZE-1; j >= 0; j--) sum += a[j];
}
```

1) How would we compute AMAT if we had the local L1 miss rate ("L1Miss"), the local L2 miss rate ("L2Miss") and the memory access time ("Memory")? Use "H1" and "H2" to represent the L1 and L2 hit times respectively. (We will compute these quantities later in the question)

M2-4: (continued)

2) For the program above, express the local miss rate for the L1 cache in general terms as a function of the L1 cache size (write L1 for the size of L1 in bytes). Hint: The miss rate is 0 for a 1 MiB cache, 0.5 for a 0.5 MiB cache and 1 for a 0 MiB (i.e., no) L1 cache.

3) What is the global miss rate for the L2 cache as a function of the L1 cache size? Hint: Start by expressing the global miss rate as a function of the L2 cache size.

4) What is the local miss rate for the L2 cache as function of the L1 and L2 sizes? Hint: Use your results from questions 2 and 3.

5) Assume the hit time of the L1 cache is 10 cycles, the hit time of the L2 cache is 20 cycles and the memory access time is 100 cycles. Using the formula from question 1, what is the AMAT for this system as a function of only the L1 size?

6) What sizes of L1 and L2 caches should we pick to minimize the AMAT? (assume the caches have non-zero size, i.e., both of them exist)

SID: _____

F1: *Paging all CS61C students* (9 points)

Consider a byte-addressed machine with a 13-bit physical address space that can hold two pages in memory. Every process is given 16MiB of virtual memory and pages are evicted with an LRU replacement scheme.

a) What are the sizes of the following fields in bits?

Virtual Page Number: _____ Virtual Address Offset: _____

Physical Page Number: _____ Physical Address Offset: _____

b) Consider the following code snippet:

```
// a and b are both valid pointers to
// different arrays of length ARRAY_SIZE
void enumerate(int* a, int* b) {
     for (int i = 0; i < ARRAY_SIZE; i++) {
          a[i] = i;
          b[i] = ARRAY_SIZE - i;
     }
}
```

The compiled binary for the program containing this code snippet weighs in at 4096B. If this code was executed on the machine, what is the maximum value of `ARRAY_SIZE` that would allow this code to execute with 0 page faults in the best-case scenario? (Answer in IEC prefix: 8Gi, 32Ti, etc)

`ARRAY_SIZE` = _____

c) How could we modify the above code snippet to allow a larger `ARRAY_SIZE` and execute with the fewest page faults in the best-case scenario? Write the new code below:

SID: _____

F2: *Why can't you use parallelism at a gas station? It might cause a spark.* (10 points)

1. Optimize `factorial()` using SIMD intrinsic(AVX).

```
double factorial(int k) {
  int i;
  double f = 1.0;
  for (i = 1 ; i <= k ; i++) {
    f *= (double) i;
  }
  return f;
}
```

You might find the following intrinsics useful:

`__m256d _mm256_loadu_pd(double *s)`	returns vector(s[0], s[1], s[2], s[3])
`void _mm256_store_pd(double *s, __m256d v)`	stores p[i] = v_i where i = 0, 1, 2, 3
`__m256d _mm256_mul_pd(__m256d a, __m256d b)`	returns vector(a_0b_0, a_1b_1, a_2b_2, a_3b_3)

```
double factorial(int k) {
  int i, j;
  double f_init[] = {1.0, 1.0, 1.0, 1.0};
  double f_res[4];
  double f = 1.0;
  // initialize f_vec
  __m256d f_vec = ____________________________________________________________

  // vectorize factorial
  for (i = 1 ; i <= ____________________________ ; ____________________________) {
    double l[] = {
      (double) ____________________________, (double) ____________________________,

      (double) ____________________________, (double) ____________________________};

    __m256d data = ____________________________________________________________

    ____________________________________________________________________   }
  // reduce vector


  ____________________________
  for (j = 0 ; j < 4 ; j++) {
    f = ____________________________________________________________________;
  }
  // handle tails
  for ( ; i <= k ; i++) {
    ____________________
  }
  return f;
}
```

F2: (continued)

2. Cache Coherence:
We are given the task of counting the number of even and odd numbers in an array, A, which only holds integers greater than 0. Using a single thread is too slow, so we have decided to parallelize it with the following code:

```
#include <stdio.h>
#include "omp.h"
void count_eo (int *A, int size, int threads) {
    int result[2] = {0, 0};
    int i,j;

    omp_set_num_threads(threads);

    #pragma omp parallel for
    for (j=0; j<size; j++)
        result[(A[j] % 2 == 0) ? 0 : 1] += 1;

    printf("Even: %d\n", result[0]);
    printf("Odd: %d\n", result[1]);

}
```

As we increase the number of threads running this code:

a) Will it print the correct values for Even and Odd? If not, explain the error.

b) Can there be false sharing if the cache block size is 8 bytes?

c) What about 4 bytes?

SID: _____

F3: *This isn't a bathroom. Why is there potpourri?* (10 points)

1. **T/F Questions (Circle one. If the circling is unclear, you will receive no credit.)**

1) CPUs need separate instructions to access I/O devices. (True / False)
2) Segmentation (base + bound) has fragmentation problems. (True / False)
3) Exceptions in early pipeline stages override exceptions in later stages for a given instruction. (True / False)
4) Exceptions are handled in the pipeline stage where they occur. (True / False)

2. **Polling, Interrupts, and DMA**

1) Choose polling or interrupt for the following devices.

Device	Data Rate	Transfer Block Size	Polling/Interrupt?
A	80B/s	4B	
B	400MB/s	4B	
C	400MB/s	2KB	

2) To support interrupts, the CPU should be able to save and restore the current state. Which of the following should be saved before handling interrupts to ensure correct execution?

a. Program Counter b. User Registers c. TLB d. Caches

3) To which device in 1) is direct memory access (DMA) most beneficial? Explain briefly.

3. **Warehouse Scale Computing and Amdahl's Law.**

1) We are going to train convolutional neural networks on Amazon EC2. It turns out that 90% of training can be parallelized but the rest takes twice as long due to the communication overhead among the instances. What is the maximum speedup we can achieve?

2) Which of the following can increase the maximum speedup in 1)?
a. Use more instances
b. Deploy the application across multiple arrays.
c. Reduce the number of reduce operations.
d. Increase network bandwidth.
e. Increase the capacity of disks.

F3: (continued)

4. **Hamming Error-Correction Code (ECC)**

1) Suppose we have a one-byte data value, 01101101_{two}. Fill in the encoded data in the following table.

Bit	1	2	3	4	5	6	7
Encoded Data							
P1	X		X		X		X
P2		X	X			X	X
P4				X	X	X	X

2) Assume that we have an encoded value, 1001110_{two} with a single-bit error. Indicate below each parity bit if it has an error:

Parity Bit	P1	P2	P4
OK/ERROR			

Incorrect bit position: _____

Correct data: _____

5. **Dependability and RAID**

1) Which of the following can increase the availability?

a. Increasing MTTF
b. Decreasing MTTF
c. Increasing MTTR
d. Decreasing MTTR
e. Redundant data copies

2) Explain very briefly why RAID1 is the most expensive form of RAID.

3) How many check disks are needed for RAID3?

4) Explain why RAID5 has a higher write throughput than RAID4.