

15-213 Introduction to Computer Systems
Final Exam
May 10, 2007

Name: Model Solution
Andrew User ID: fp
Recitation Section: _____

- This is an open-book exam.
- Notes and calculators are permitted, but not computers.
- Write your answer legibly in the space provided.
- You have 180 minutes for this exam.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat [edu_assist_pro](#)

			ore
			0
Assembly Language	2		
Optimization	3	20	20
Cache Memory	4	20	20
Signals	5	20	20
Garbage Collection	6	20	20
Threads	7	20	20
Synchronization	8	20	20
Total		150+10	160

1. Floating Point (20 points)

In this problem we consider properties of floating point operations. For each property state whether it is true or false. If false, give a counterexample as a (possibly negative) power of 2 within the range of precision for the variables. We assume that the variables on an x86_64 architecture are declared as follows

```
float x, y, z;
double d, e;
```

and initialized to some unknown value **different from NaN**, $+\infty$, and $-\infty$. We have given the first answer as an example.

$(x + y) + z == x + (y + z)$	false	$x = 1, y = 2^{127}, z = -2^{127}$
If $x > 0$ then $x / 2 > 0$	false	$x = 2^{-149}$
$(x + y) * z == x * z + y * z$	false	$x = 2^{127}, y = -2^{127}, z = 2^{127}$
If $x \geq y$ and $z \leq 0$ then $x * z \leq y * z$	true	
If $x > y$ then (
If $d > e$ then		$d = 2^{129}, e = 2^{128}$
$x + d < x$		$x = -2^{127}$

2. Assembly Language (20 points)

In this problem we consider an illustrative program for multiplication of two unsigned int's, returning an unsigned long int holding the product.

```
unsigned long mult (unsigned i, unsigned k) {
    unsigned long p = 0;
    unsigned long q = k;
    while (i != 0) {
        if (i & 1)
            p = p + q;
        q = q << 1;
        i = i >> 1;
    }
    return p;
}
```

The following is the resulting machine code when compiled on an x86_64 machine with gcc -O2, omitting two instructions.

```
mult:
    xorl    %ecx, %e
    mov     %esi, %edx
    testl   %edi
    jmp     .L8
.L10:
    leaq    %rdx, %rdx, %rax
    testb   $1, %dil

    _____ # missing conditional move
    addq    %rdx, %rdx
    shrl    %edi
.L8:
    jne     .L10

    _____ # missing move
    ret
```

1. (5 pts) For each register, give the value it holds during the iteration, expressed in terms of the C program.

Register	C expression
%rcx	p
%rdx	q
%rax	p+q
%edi	i
%dil	(char) i

2. (5 pts) Fill in the missing two instructions in the code.

```
cmovne %rax, %rcx and movq %rcx, %rax
```

3. (4 pts) Rewrite the loop to use a conditional jump instead of a conditional move.

Assignment Project Exam Help
See one solution below; there are many others.

```
testb $1, %dil  
jne .L9  
movq %rax, %rcx  
.L9  
addq %rdx, %rdx
```

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

4. (3 pts) Explain briefly why the compiler preferred to use a conditional move instruction.

Because the branch misprediction penalty would make the loop slower, especially since the outcome of test will be difficult to accurately predict.

5. (3 pts) Assume we declared and initialized

```
int i, k;  
long m;
```

and called

```
m = (long)mult((unsigned)i, (unsigned)k);
```

using the above definition of `mult`. Will `m` hold the correct value of the signed product of `i` and `k`? Circle the correct answer.

yes no **no**

Briefly explain your answer.

For example, when multiplying 1 times -1 , the negative 1 will actually be interpreted as $2^{32} - 1$ and the result will also be $2^{32} - 1$ instead of -1 . However, the answer will be correct modulo 2^{32} because on two's-complement representations, signed integers operate in the range $[-2^{w-1}, 2^{w-1}-1]$ where w ($= 32$, in this case).

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

3. Optimization (20 points)

Consider the following code for calculating the dot product of two vectors of double precision floating point numbers.

```
double dot_prod(double A[], double B[], int n) {
    int i;
    double r = 0;
    for (i = 0; i < n; i++)
        r = r + A[i] * B[i];
    return r;
}
```

Assume that multiplication has a latency of 12 cycles and addition a latency of 7 cycles and load 4 cycles. Also assume that there are an unlimited number of functional units. [Hint: Under this assumption, theoretically optimal performance is dominated by the critical data dependency path.]

1. (5 points) What is the theoretically optimal CPE for this loop?

7 CPE, since the addition constitutes the critical path.

2. (10 points) Show that the commutativity of addition and commutativity of multiplication are insignificant. y
ding errors

```
double dot_prod2(double A[], double B[], int n) {
    int i;
    double r = 0;
    for (i = 0; i < n-1; i+=2)
        r = r + (A[i] * B[i] + A[i+1] * B[i+1]);

    for (; i < n; i++)
        r = r + A[i] * B[i];

    return r;
}
```

3. (5 points) What is the theoretically optimal CPE for this loop?

$7/2 = 3.5$ CPE, since the critical path is still addition, but now two elements will be added in each iteration.

4. Cache Memory (20 points)

In this problem we explore the operation of a basic TLB as a cache. Assume the following

- Virtual addresses are 32 bits.
- The virtual page number (VPN) is 24 bits.
- The physical page number (PPN) is 32 bits.
- The TLB is 2-way set associative containing a total of 512 lines.

1. (6 points) Please fill in the following blanks by giving a bit range, such as “0–15”.

- The VPO of a virtual address consists of bits 0–7 of the VA.
- The VPN of a virtual address consists of bits 8–31 of the VA.
- The PPO of a physical address consists of bits 0–7 of the PA.
- The PPN of a physical address consists of bits 8–39 of the PA.
- The TLB index (TLBI) consists of bits 0–7 of the VPN.
- The TLB tag (TLBT) consists of bits 8–23 of the VPN.

We show a part of t

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

3D	1	0x08	
		0x08	
3E	0	0xF3	
	1	0x083F	0xAB18ED24
3F	1	0x409A	0x0913ABDE
	1	0x083F	0xAB18ED24
40	0	0x083E	0x0913ABDE
	1	0x3E40	0xAB18ED24

2. (7 points) Assume the virtual address is $0x083F3E9A$. Fill in the following table in hexadecimal notation. Write **U** for any value that is unknown, that is, not determined from the parameters and the table above.

Parameter	Value
VPN	$0x083F3E$
VPO	$0x9A$
TLBI	$0x3E$
TLBT	$0x083F$
Cache Hit? (Y/N/U)	Y
PPN	$0xAB18ED24$
PA	$0xAB18ED249A$

3. (7 points) Assume the virtual address is $0x083E409B$. Fill in the following table in hexadecimal notation. Write **U** for any value that is unknown, that is, not determined from the p

VPN	0
VPO	0
TLBI	$0x40$
TLBT	$0x083E$
Cache Hit? (Y/N/U)	N
PPN	U
PA	U

5. Signals (20 points)

Consider the following program.

```
int counter = 0;

void handler (int sig) {
    counter++;
}

int main() {
    signal(SIGUSR1, handler);
    signal(SIGUSR2, handler);
    int parent = getpid();
    int child = fork();

    if (child == 0) {

        /* insert code here */
        exit(0);
    }

    sleep(1);
    waitpid(child, NULL, 0);
    printf("Received %d USR{1,2} signals\n", counter);
    return 0;
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

For each of the following four versions of the above code, list the possible outputs of this program, assuming that all function and system calls succeed and exit without error. You may also assume no externally issued signals are sent to either process.

1. (5 pts)

```
kill(parent, SIGUSR1);
kill(parent, SIGUSR1);
```

1,2: If the second SIGUSR1 is sent before the first one is received it will be dropped.

2. (5 pts)

```
kill(parent, SIGUSR1);  
kill(parent, SIGUSR1);  
kill(parent, SIGUSR1);
```

1,2,3: The second and third SIGUSR1 may be sent before the first one is received.

3. (5 pts)

```
kill(parent, SIGUSR1);  
kill(parent, SIGUSR2);
```

1,2: Because of a race condition when SIGUSR2 is received while SIGUSR1 is handled, one increment may be dropped.

4. (5 pts)

```
kill(paren  
kill(paren  
kill(parent, SIGUSR1);  
kill(parent, SIGUSR2);
```

1,2,3,4: Two consecutive occurrences as in the answer to the previous question can lead to answers 1+1, 1+2, 2+1 or 2+2. And the race condition from the previous question can lead to the answer 1 if the first three signals are sent before any are received.

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

6. Garbage Collection (20 points)

In this problem we consider a tiny list processing machine in which each memory word consists of two bytes: the first byte is a pointer to the tail of the list and the second byte is a data element. The end of a list is marked by a pointer of 0x00. We assume that the data element is never a pointer.

We start with the memory state on the left, where the range 0x10–0x1F is the from-space and the range 0x20–0x2F is the to-space. All addresses and values in the diagram are in hexadecimal.

Write in the state of memory after a copying collector is called with root pointers 0x10 and 0x12, in this order. You may leave cells that remain unchanged blank.

Please be sure to use the proper breadth-first traversal algorithm covered in lecture.

Before GC			After GC					
Addr	Ptr	Data	Addr	Ptr	Data	Addr	Ptr	Data
10	14	A2	10	20		20	24	A2
12	1A	1F	12	22		22	16	1F
14	1E	02	14	24		24	28	02
16	1E						2A	BC
18	00						2C	8F
1A	18	BC	1A	26			00	33
1C	12	DE	1C					
1E	10	8F	1E	28		2E		

After garbage collection, free space starts at address 2C

7. Threads (20 points)

Consider three concurrently executing threads in the same process using two semaphores s_1 and s_2 . Assume s_1 has been initialized to 1, while s_2 has been initialized to 0.

What are the possible values of the global variable x , initialized to 0, after all three threads have terminated?

```
/* thread A */  
P(&s2);  
P(&s1);  
x = x*2;  
V(&s1);
```

```
/* thread B */  
P(&s1);  
x = x*x;  
V(&s1);
```

```
/* thread C */  
P(&s1);  
x = x+3;  
V(&s2);  
V(&s1);
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

The possible sequences are B,C,A ($x = 6$) or C,A,B ($x = 36$)

Add WeChat edu_assist_pro

8. Synchronization (20 points)

We explore the so-called *barbershop problem*. A barbershop consists of a n waiting chairs and the barber chair. If there are no customers, the barber waits. If a customer enters, and all the waiting chairs are occupied, then the customer leaves the shop. If the barber is busy, but waiting chairs are available, then the customer sits in one of the free chairs.

Here is the skeleton of the code, without synchronization.

```
extern int N;    /* initialized elsewhere to value > 0 */
int customers = 0;
```

```
void* customer() {
```

```
    if (customers > N) {
```

```
        return NULL;
```

```
    }
```

```
    customers += 1;
```

```
    getHairCut();
```

```
    customers -= 1;
```

```
    return NULL;
```

```
}
```

```
void* barber() {
```

```
    while(1) {
```

```
        cutHair();
```

```
    }
```

```
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

For the solution, we use three binary semaphores:

- `mutex` to control access to the global variable `customers`.
- `customer` to signal a customer is in the shop.
- `barber` to signal the barber is busy.

1. (5 points) Indicate the initial values for the three semaphores.

- `mutex`
- `customer`
- `barber`

2. (15 points) Complete the code above filling in as many copies of the following commands as you need, but no other code.

```
P(&mutex);  
V(&mutex);  
P(&customer);  
V(&customer);  
P(&barber);  
V(&barber);
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Solution: There are a number of solutions; below is one. Be careful to release the mutex before leaving. For this solution, initial values are `mutex = 1` (variable `customers` may be accessed), `customer = 0` (no customers) and `barber = 0` (barber is not busy).

```
void* customer() {  
  
    P(&mutex);  
    if (customers > N) {  
        V(&mutex);  
        return NULL;  
    }  
    customers += 1;  
    V(&mutex);  
  
    V(&customer);  
    P(&barber);  
    getHairCut();  
  
    P(&mutex);  
    customers += 1;  
    V(&mutex);  
  
    return NULL;  
}  
  
void* barber() {  
    while(1) {  
        P(&customer);  
        V(&barber);  
        cutHair();  
    }  
}
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro