Все замечания, исправления и другие fix-ы приветствуются!
Не рекомендую пользоваться, если не понимаете, почему здесь такие цифры/картинки/слова.

Тема-1

1. **Какие аппаратные механизмы необходимы для организации мультипрограммного режима? Как обеспечить мультипрограммный режим без этих механизмов? Как обеспечить, если отсутствует только один из этих механизмов?**

Требуются: защита памяти, прерывания, привилегированный режим, таймер.
Если нет таких механизмов, то можно использовать виртуальную машину - единый интерпретатор, исполняющий программы пользователей и обеспечивающий эмуляцию всех этих механизмов.
Видимо отсутствие одного из этих механизмов равносильно отсутствию всех.

Возможно также выполнять на машине без этих требований программу, созданную специальным компилятором, который всегда при создании машинного кода вставляет дополнительные инструкции и проверки для эмуляции мультипрограмного режима. Этот способ эффективнее полной эмуляции, хотя и сложнее на создание компилятора.

Тема-2

1. **Имеется механизм двоичных семафоров. Опираясь на него, реализуйте P-операцию и V-операцию для общего (считающего) семафора. Активное ожидание освобождения семафора не допускается**.

В приведенном алгоритме возникают интересные вопросы, связанные с выделением памяти.

```
struct dsemaphore{
public:
        bsemaphore access;
        bsemaphore wait;
        dsemaphore(semName_t name,int N){s=N;access=1;wait=1;}  // создание владельцем семафора
        dsemaphore(semName_t name){ }                           //Присоединение к семафору.
        void P();
        void V();
private:
        int s;
};

void dsemaphore::P(){
        P(wait);
        P(access);
        s--;
        if(s>0)   V(wait);
        V(access);
}
void dsemaphore::V(){
        P(access);
        s++;
        if(s==1){
                V(wait);
        }
        V(access);
}
```

2. **Имеется команда TSL и команда объявления прерывания указанному процессору. Опираясь на них, реализуйте на мультипроцессоре P-операцию и V-операцию для двоичного семафора. Активное ожидание освобождения семафора не допускается.**

```
Init(s)
Tbl[s][0]=2
Tbl[s][1]=2

Restart:
        sleep(rnd())
P(s):   tsl r0, syn
        cmp r0,1
        jz restart

V(s):   tsl r0, syn
        cmp r0,1
        jz V
        inc tbl[s][1]
        cmp tbl[s][1], tbl[s][0]
```

	mov tbl[s][tbl[s][0]], pid
	inc tbl[s][0]
	cmp tbl[s][0], tbl[s][1]+1
	jz own
	mov syn, 0
	sleep
	ret
	own: mov syn, 0
	ret
		jz exit
		wake tbl[s][tbl[s][1]]
		exit: mov syn, 0
		ret

3. **Имеется механизм двоичных семафоров. Опираясь на него, реализуйте операторы POST(имя переменной-события) и WAIT(имя переменной-события). Активное ожидание события не допускается.**

struct event{
semaphore access;
semaphore waiters[N];
int nWaiters;
int state;
}
POST(event *ev){
P(ev->access);
ev->state = 1;
for(int i=0;i<nWaiters;i++){
V(ev->waiters[i]);
}
V(ev->access);
}
	WAIT(event *ev){
	P(ev->access);
	if(ev->state == 1){
	V(ev->access);
	}else{
	int index = ev->nWaiters++;
	P(ev->waiters[index]);
	V(ev->access);
	P(ev->waiters[index]);
	V(ev->waiters[index]);
	}
	}

s=0; POST(s){ V(s); }	WAIT(){ P(s);V(s); }

4. **Оцените, во сколько раз нижеприведенный алгоритм метода последовательной верхней релаксации можно выполнить быстрее, чем последовательный, если число процессоров мультипроцессора = N, время выполнения одного оператора присваивания (*A[i][j]=....*) равно 1, временами выполнения остальных операторов можно пренебречь.**

```
float  A[ L1 ][ L2 ];

semaphore s[ L1 ][ L2 ]; /*  массив двоичных семафоров с нулевым начальным значением */

for ( j = 0; j < L2; j++)   { post( s[ 0 ][ j ]) }

parfor ( i = 1; i < L1-1; i++)

        for ( j = 1; j < L2-1; j++)

          {   wait( s[ i-1 ][ j ]);

              A[ i ][ j ] = (A[ i-1 ][ j ] + A[ i+1][ j ] + A[ i ][ j-1 ] + A[ i ][ j+1 ]) / 4;

              post( s[ i ][ j ]);

          }
```

Будем считать, что как минимум L2 > N+2
Время последовательного выполнения: Tseq = (L1-2)*(L2-1)
Время параллельного выполнения:
Произойдёт (L1 div N) параллельных циклов с полной загрузкой процессоров
Произойдет (L1 mod N) параллельных циклов с неполной загрузкой.
Задержка завершения последнего процесса составит
Если полная загрузка процессоров {(L1 mod N)==0} – N /2 шагов
Если полная загрузка (L1 mod N)/2 шагов
Общее время выполнения:

(L1 div N)*(L2-2)+(L1 mod N)*(L2-2)+ (((L1 mod N)==0)?N/2: (L1 mod N)/2)

Итого соотношение:
(L1-2)*(L2-1)/(L1 div N)*(L2-2)+(L1 mod N)*(L2-2)+ (((L1 mod N)==0)?N/2: (L1 mod N)/2)

Замечание: Если L2<N+2, тогда добавление процессоров сверх числа L2-2 не даст ни какого выигрыша в производительности по сравнению с N=L2-2.

5. Имеется механизм двоичных семафоров. Опираясь на него, реализуйте задачу читателей и писателей (алгоритмы предоставления прав доступа процессам-читателям и процессам-писателям):

Процесс-писатель должен получать исключительный (монопольный) доступ к базе данных (других писателей или каких-либо читателей быть не должно). Произвольное число процессов-читателей может работать одновременно, но любой читатель может получить доступ только при отсутствии работающих писателей.

Запросы на доступ должны удовлетворяться "справедливо" - в порядке их поступления (можно исходить из "справедливости" удовлетворения запросов на двоичные семафоры).

```
semaphore reader;
semaphore writer;
semaphore access;
int rd=0;
```

```
Write_enter(){
        P(writer);
        P(reader);
}
```

```
Write_leave(){
        V(reader);
        V(writer);
}
```

```
Read_enter(){
        P(writer);
        P(access);
        rd++;
        if(rd==1)P(reader);
        V(access);
        V(writer);
}
```

```
Read_leave(){
        P(access);
        rd--;
        if(rd==0)
            V(reader);
        V(access);
}
```

6. Какие модели консистентности памяти удовлетворяют алгоритму Деккера (алгоритм без каких-либо изменений будет работать правильно), а какие нет? Объясните ответ.

```
void enter_region( int i )
{
    try:  flag[ i ]=TRUE;
    while (flag [( i+1 ) % 2]){
        if ( turn = = i ) continue;
        flag[ i ] = FALSE;
        while ( turn != i );
        goto try;
      }
}
```

```
void leave_region( int i )
{
    turn = ( i +1 )  % 2;
    flag[ i ] = FALSE;
}
```

Модель	
Строгая	Да, так как данные полностью согласованы.
Последовательная	Да, так как процессоры видят записи в едином порядке,
Причинная	Нет, так как процессы независимо пишут в память. Причинности между записями не имеется. Процессы могут не увидеть записи, сделанной на другом процессоре, и войти в КС.
PRAM	Нет, так как ещё слабее, чем последовательная. Тот же пример.
Процессорная	Считая переменные flag[0] и flag[1] разными при процессорной консистентности алгоритм работать не будет.
Слабая	Нет, так как необходимо выделение синхронизационных переменных.

При любой модели консистентности с синхронизацией алгоритм работать не будет.

7. Какие модели консистентности памяти удовлетворяют алгоритму Петерсона (алгоритм без каких-либо изменений будет работать правильно), а какие нет? Объясните ответ.

```
void enter_region( int i )
{
    int other=1- i;                    /* номер другого процесса */
    flag[ i ] = TRUE;
    turn = i;
    while (turn = = i  &&  flag[ other ] = =  TRUE)   /* пустой оператор */;
}
void leave_region( int i ){              flag[ i ] =  FALSE;          }
```

Модель	
Строгая	Да, так как данные полностью согласованы.
Последовательная	Да. Так как все процессоры видят модификации данных в едином порядке (включая и автора изменений)
Причинная	Нет. Причинные связи отслеживаются по чтению, предшествующему записи. Здесь наоборот. Запись предшествует чтению, что не будет отслежено.
PRAM	Нет, так как каждый процессор сможет изменить значения разделяемых переменных в локальной памяти и войти в КС.
Процессорная	Считая переменные flag[0] и flag[1] разными при процессорной консистентности алгоритм работать не будет, так как оба процессора могут войти в КС.
Слабая	Нет, так как необходимо выделение синхронизационных переменных.

При любой модели консистентности с синхронизацией алгоритм работать не будет.

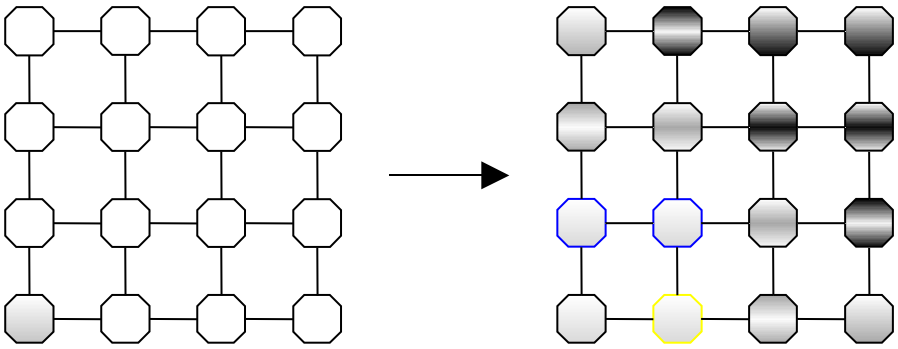
Тема-3

1. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо выполнить операцию передачи сообщения длиной N байт всем процессам от одного (MPI_BCAST) - процесса с координатами (0,0). Сколько времени потребуется для этого, если все процессы выдали ее одновременно. Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Pr-0	A0			
Pr-1				
Pr-2				
Pr-3				

BROADCAST ➔

A0			
A0			
A0			
A0			



8*(Ts+N*Tb)

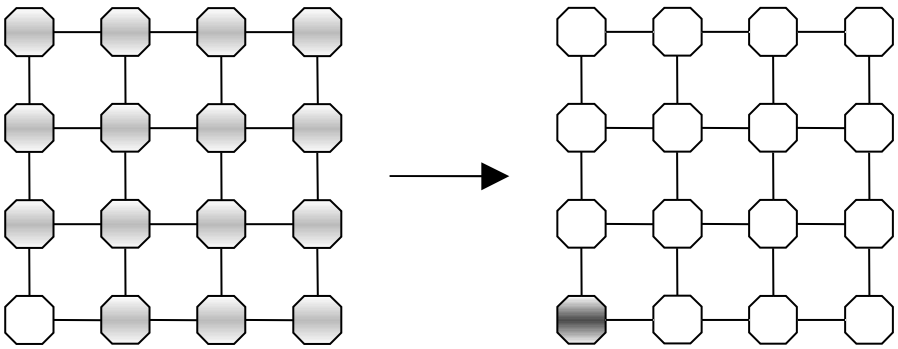
2. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо выполнить операцию сбора данных от всех процессов (длиной один байт) для одного (MPI_GATHER) - процесса с координатами (0,0). Сколько времени потребуется для этого, если все процессы выдали ее одновременно. Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Pr-0	A0	A1	A2	A3
Pr-1				
Pr-2				
Pr-3				

SCATTER ➔

GATHER ➔

A0			
A1			
A2			
A3			



8*(Ts+Tb*Lm) – время на извещение.
8*(Ts+Tb*N) – время требуемое для получения данных при сбалансированной загрузке.
Возможна оптимизация: первые сообщения посылаются ближайшим транспьютерам:
Они начнут сразу передавать данные. Дальше при больших N сообщения посылаются самым дальним,
И по мере удаления. Цель – сохранить оба входных канала приема максимально заполненными.
Тогда 8*(Ts+N*Tb) +Ts+Lm.

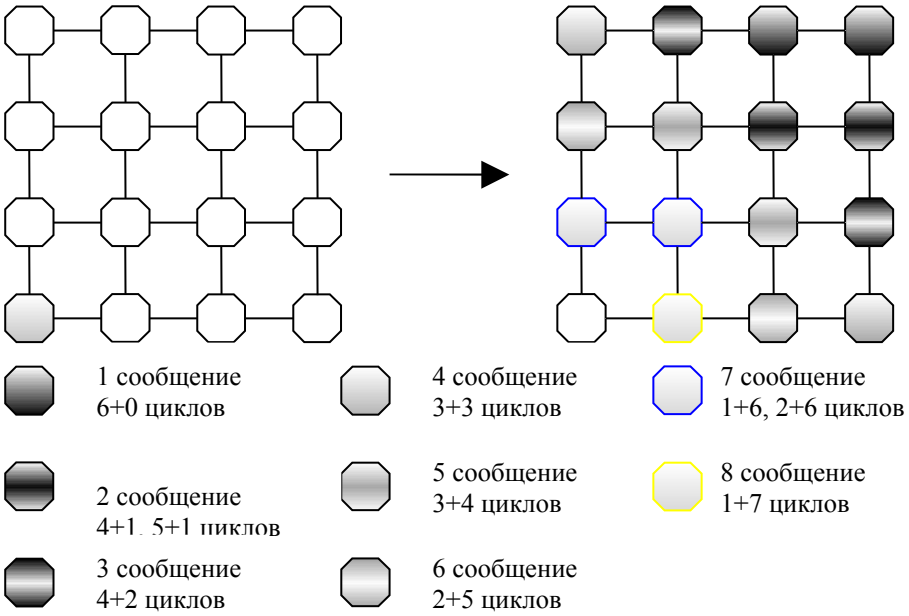
Если N не достаточно велико, то есть будет простой приема при такой схеме (например, N<100), подсчет усложняется. Усложняется также выбор оптимального порядка рассылки сообщений.

3. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо выполнить операцию рассылки данных (длиной один байт) всем процессам от одного (MPI_SCATTER) - процесса с координатами (0,0). Сколько времени потребуется для этого, если все процессы выдали ее одновременно. Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Pr-0	A0	A1	A2	A3
Pr-1				
Pr-2				
Pr-3				

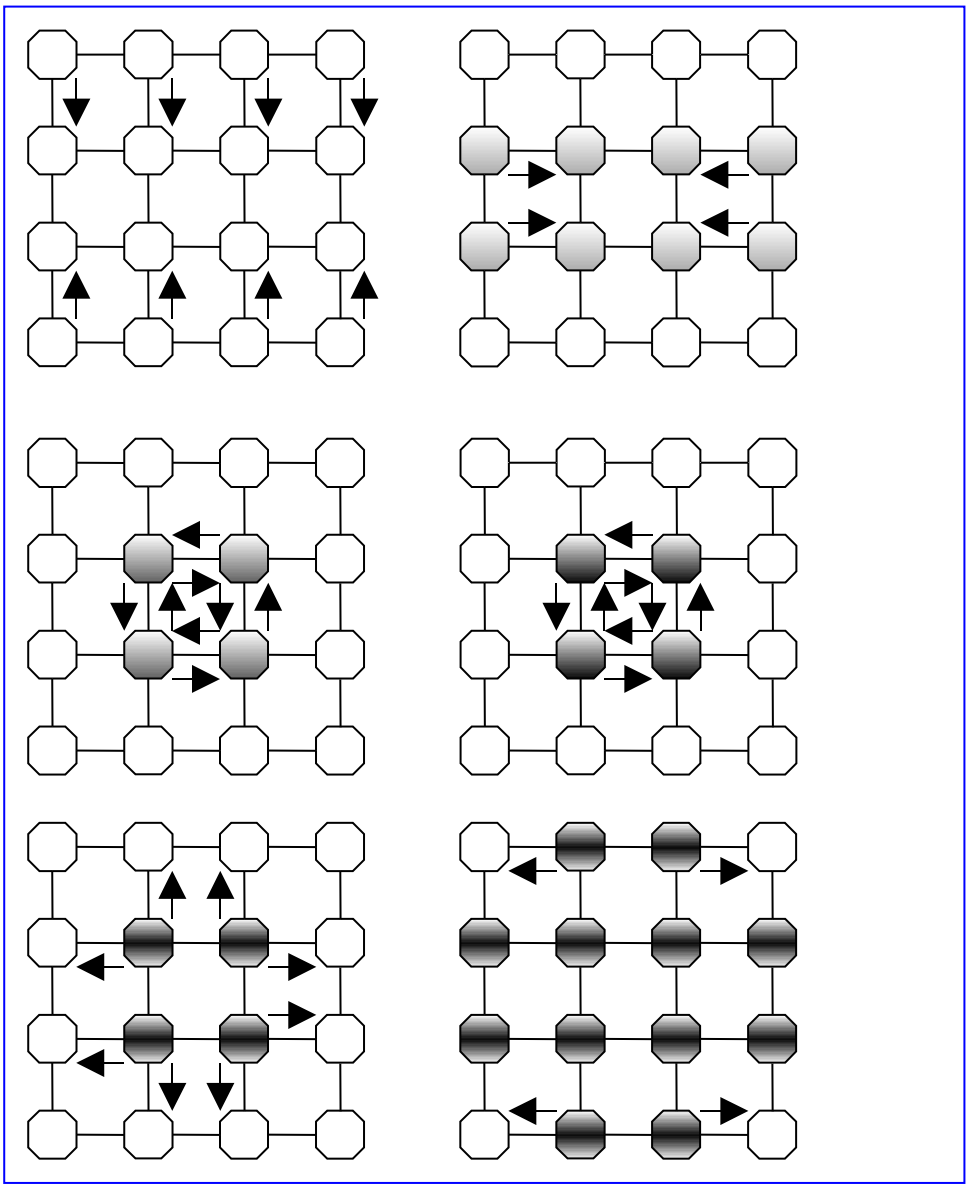
SCATTER ➔
GATHER ➔

A0			
A1			
A2			
A3			

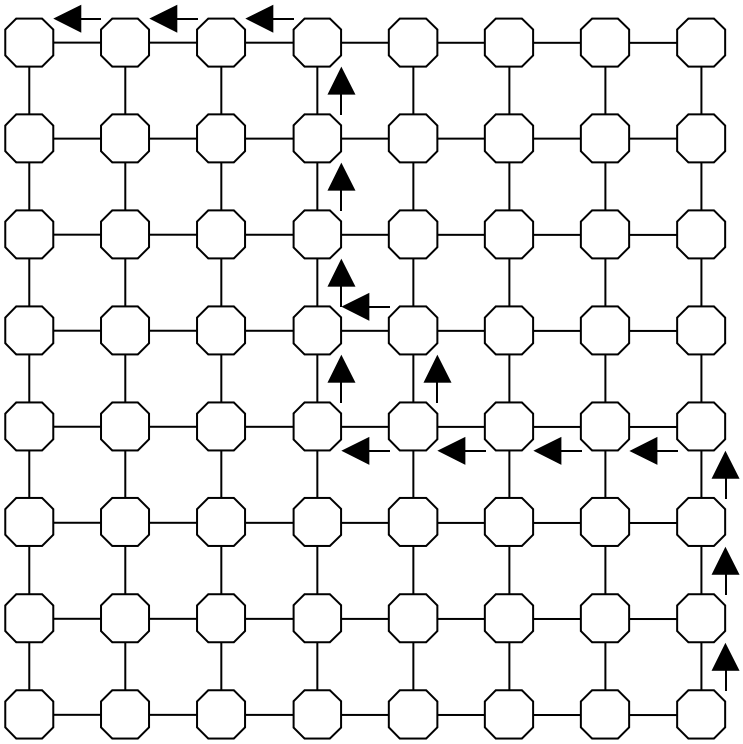


Итого данные будуь получены через 8 циклов, т.е 8*(Ts+N*Tb)

4. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо выполнить операцию нахождения максимума среди 16 чисел (каждый процесс имеет свое число). Сколько времени потребуется для получения всеми максимального числа, если все процессы выдали эту операцию редукции одновременно. А сколько времени потребуется для нахождения максимума среди 64 чисел в матрице 8*8? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**



T = 6*(Ts+L*Tb)



14*(Ts+L*Tb)

5. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо переслать очень длинное сообщение (длиной L байт) из узла с координатами (0,0) в узел с координатами (3,3). Сколько времени потребуется для этого, если передача сообщений точка-точка выполняется в буферизуемом режиме MPI? А сколько времени потребуется при использовании синхронного режима и режима готовности? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Замечание. MPI находится на более высоком уровне, чем транспьютерная решётка (в модели OSI).
Собственно буферизуемый или не буферизуемый режим MPI влияет только на процесс-отправитель и процесс-получатель.
Однако транспьютеры могут работать в режиме передачи сообщений с буферизацией или без неё. При режиме с ьуферизацией транспьютер не имеет право инициировать посылку сообщения в выходной канал до того, как он получил последний байт с входного канала.
При отсутствии буферизации транспьютер должен сразу при получения первого байта сообщения инициировать пересылку по выходному каналу (если узел не является получателем). Считаем, что транспьютер имеет возможность инициировать пересылку во исходящему каналу сразу после начала передачи. В общем случае следует учитывать задержку на прием служебной информации с адресом получателя. Существуют системы, в которых такая задержка строго равна 1 байту, т.е. первый байт кадра сообщает коммутатору в какой выходной канал следует направлять все сообщение.

Также учтем, что само сообщение можно распараллелить на 2 выходных канала, что отразим сокращением длины в 2 раза.

Буферизуемый режим транспьютеров – Ts*6+Tb*(L/2)*6
Небуферизуемый режим транспьютеров – Ts*6+(L/2)* Tb+6

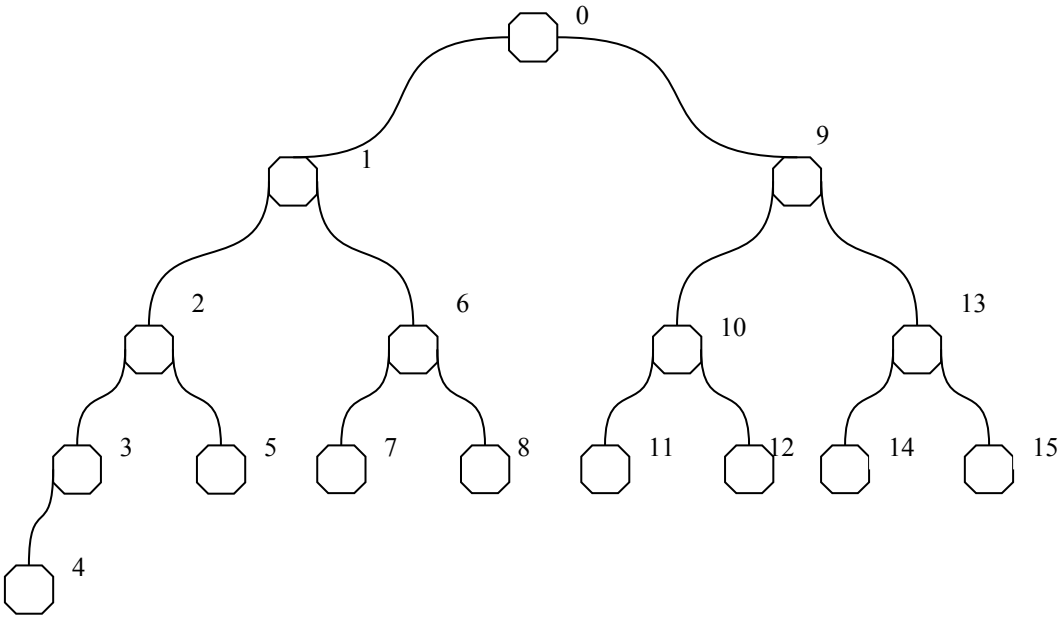
6. **В транспьютерной матрице размером 4*4, в каждом узле которой находится один процесс, необходимо переслать сообщение длиной L байт из узла с координатами (0,0) в узел с координатами (3,3). Сколько времени потребуется для этого при использовании а) неблокирующих и б) блокирующих операций MPI? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

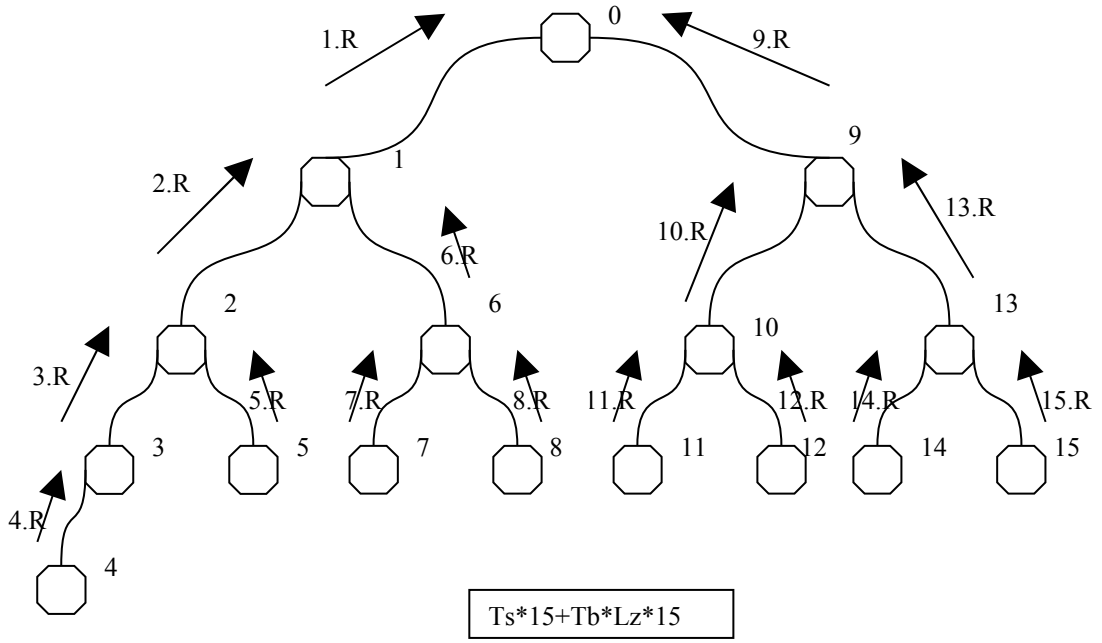
Разницы между блокирующими и неблокирующими не будет, так как блокировка относится не к подсистеме MPI, а к процессу, вызывающему.
Потребуется времени – Ts*6+Tb*L*6, если не использовать распараллеливание передачи данных.

Если использовать, то можно считать, что данные пойдут по сторонам квадрата, так как узким местом здесь будут выходные каналы узла (0,0). То есть оптимальнее не получится. Тогда получится время Ts*6+Tb*[L/2]*6.
Если при этом предположить, что промежуточные узлы могут передавать сообщение дальше, не дожидаясь полного приема сообщений, и кроме того L достаточно большое, тогда получится:
Ts*6+ Tb*[L/2]+6. Ts*6 время полной загруки конвейера, 6 – время разгрузки конвейера.

Тема-4

1. **Все 16 процессов, находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на вход в критическую секцию. Сколько времени потребуется для прохождения всеми критических секций, если используется древовидный маркерный алгоритм (маркером владеет нулевой процесс). Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**





После этого все процессы знают, куда направлять маркер, когда он придет.

0—MR—1—MR—2—MR—3—MR—4—M—3—M—2—MR—5—M—2—M—1—MR—6—MR—7—M—6—MR—8—M—6—M—1—M—0—M—9—MR—10—MR—11—M—10—MR—12—M—10—M—9—M—13—MR—14—M—13—M—15

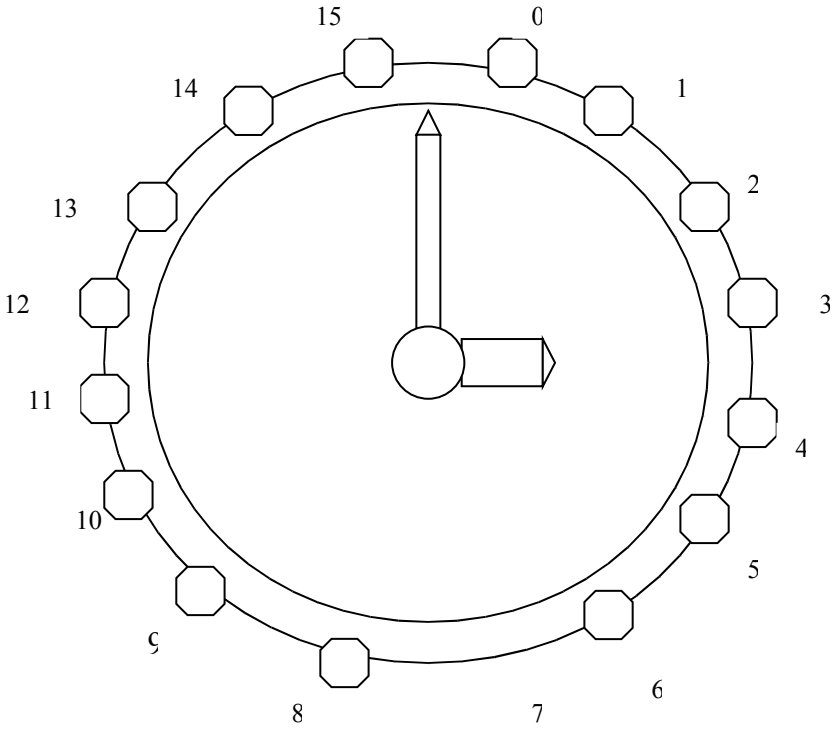
Итого сообщений передачи маркера с запросом – 12, маркера без запроса – 15, всего передач маркера – 27.

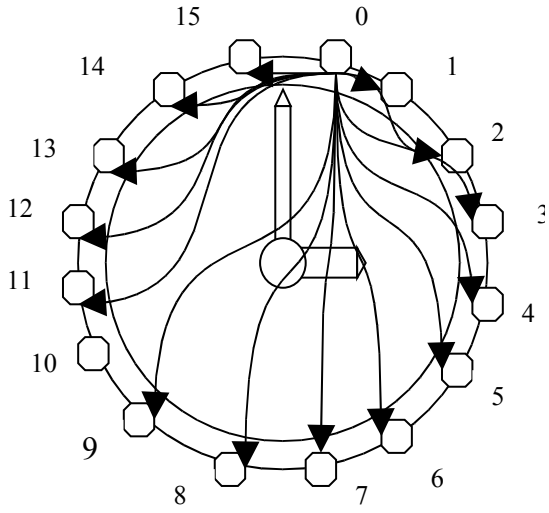
Считаем, что и маркер и маркер с запросом передаются в одном сообщении длиной в Lm

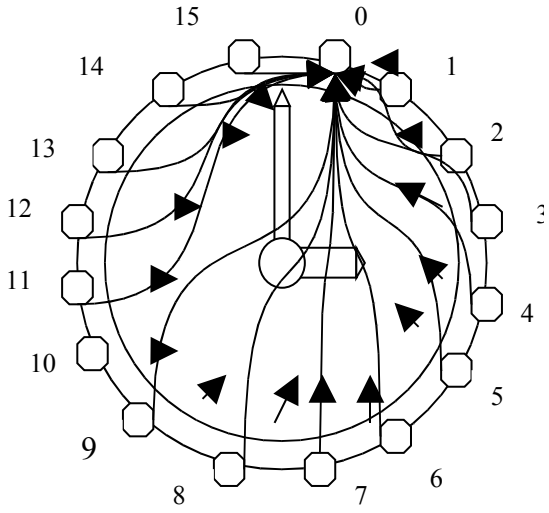
Тогда общее время = (Ts+Tb*Lz)*15+(Ts+Tb*Lm)*27.

Считая сообщение маркера очень коротким, получим 42*Ts

2. Все 16 процессов, находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на вход в критическую секцию. Сколько времени потребуется для прохождения всеми критических секций, если используется децентрализованный алгоритм с временными метками. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.







(Ts+Tb*Lm)*15

Взяв любой другой процесс, мы получим ту же картину: надо разослать 15 запросов и получить 15 ответов. При этом мы считаем, что время прохождения КС=0. Тогда получается, что надо просто умножить время получения разрешения 0-м процессом на 16.
16*15*(Ts+Tb*Lm)

Если же мы предположим, что внутри КС проводится отличное от 0 время, тогда надо будет учитывать, что некоторые сообщения не зависят от того, находится ли другой процесс в КС или нет.

3. **Все 16 процессов, находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на вход в критическую секцию. Сколько времени потребуется для прохождения всеми критических секций, если используется широковещательный маркерный алгоритм (маркером владеет нулевой процесс). Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Предположим режим с блокировкой процессов. То есть сообщение не может прийти, если не доставлены ранее отправленные.
Все кроме 0-го пошлют широковещательные запросы. Это займет 15*15*(Ts+Tb*Lz).
После этого будет передаваться только маркер, содержащий очередь запросов. Всего будет 15 передач. Каждая за время Ts+Tb*Lm.
Итого 255*(Ts+Tb*Lz)+15*(Ts+Tb*Lm).
Замечание. Нельзя отбросить Lm, так как он содержит очередь запросов и может быть достаточно большим (как минимум сравним с Ts).

4. **15 процессов, находящихся в узлах транспьютерной матрицы размером 4*4, одновременно выдали запрос на вход в критическую секцию. Сколько времени потребуется для прохождения всеми критических секций, если используется централизованный алгоритм (координатор расположен в узле 0,0)? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**
   15*3 (Ts+Tb*L)

5. **Сколько времени потребует выбор координатора среди 16 процессов, находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), если используется алгоритм**

«задиры»? «Задира» расположен в узле с координатами (0,0) и имеет уникальный номер 0. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.

6. **Сколько времени потребует выбор координатора среди 16 процессов, находящихся в узлах транспьютерной матрицы размером 4*4, если используется круговой алгоритм? Время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память считаются бесконечно быстрыми.**

Тема-5

1. Какие принципиальные решения приходится принимать при обеспечении файлового сервиса?
   Модель файла
   Можно ли модифицировать после создания
   Защита
   загрузка/разгрузка или удалённый доступ
   с состоянием / без состояния
   как обеспечить масштабируемость
   как обеспечить прозрачность
   как обеспечить высокую доступность.
   должно ли быть единое дерево ФС для всех машин
   Есть ли разница между клиентом и сервером
   ФайлСервер и Сервер директ – совпадают?
2. Интерфейс сервера директорий.
   создание /удаление директорий, именование/переименование/перемещение файлов
   создание иерархии
   единое дерево директорий или нет?
   прозрачность расположения/прозрачность миграции
   машина+путь/локальное монтирование/единое унифицированное пространство
3. Семантика разделения файлов.
   UNIX-семантика (FIFO)
   Неизменяемые файлы
   Сессии – изменения становятся видимыми при закрытии
   Транзакции
4. Серверы с состоянием и без состояния. Достоинства и недостатки.
   Серверы с состоянием. Достоинства.
   Короче сообщения (двоичные имена используют таблицу открытых файлов).
   выше эффективность (информация об открытых файлах может храниться в оперативной памяти).
   блоки информации могут читаться с упреждением.
   убедиться в достоверности запроса легче, если есть состояние (например, хранить номер последнего запроса).
   возможна операция захвата файла.
   Серверы без состояния. Достоинства.
   устойчивость к ошибкам.
   не требуется операций ОТКРЫТЬ/ЗАКРЫТЬ.
   не требуется память для таблиц.
   нет ограничений на число открытых файлов.
   нет проблем при крахе клиента.

5. Алгоритмы обеспечения консистентности кэшей в распределенных файловых системах.
   кэширование на сервере/на клиенте/диске клиента
   кэш в памяти клиента: в процессе/в ядре/кэш-менеджер
   когерентность: сквозная запись(с подслушиванием), отложенная запись, запись при закрытии, централизованное управление
6. Способы организации размножения файлов и коррекции копий.
   цели: надежность/доступность/распределение нагрузки
   явное размножение
   ленивое размножение
   симметричное размножение
   Коррекция: главная копия/голосование/

Тема-6

1. **Последовательная консистентность памяти и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует модификация 10 различных переменных 10-ю процессами (каждый процесс модифицирует одну переменную), находящимися на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания) и одновременно выдавшими запрос на модификацию. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Последовательная консистентность – все процессы видят одну и ту же последовательность записей в память.(Как будто память едина и в каждый момент времени исполняется ровно одна команда какого-то процесса).
Из лекции:
Последовательная консистентность может быть реализована также следующим образом. Страницы, доступные на запись, размножаются, но операции с разделяемой памятью не должны начинаться до тех пор, пока не завершится выполнение предыдущей операции записи, выданной каким-либо процессором, т.е. будут скорректированы все копии соответствующей страницы (все записи выполняются последовательно, блокируя на время своего выполнения работу всех процессов). В системах с упорядоченным механизмом широковещания запрос на операцию модификации памяти рассылается всем владельцам копий соответствующей страницы (включая и себя). При этом, работа процессов не блокируется. Одним процессором могут быть выданы несколько запросов на модификацию данных. Любая операция чтения не должна выполняться до того как будут выполнены все выданные данным процессором запросы на модификацию (процессор получит и выполнит «свои» запросы).

Не из лекции: с процессом-координатором.

Реализация последовательной консистентности в DSM с полным размножением:

а) при записи:
- Посылается координатору запрос на модификацию.
- Координатор принимает запрос на модификацию, присваивает ей номер и высылает автору номер.Если координатор заведомо знает, что процесс не имеет каких-то модификаций, от высылает их сразу в этом ответе.
- Приняв номер автор должен получить недостающие модификации (если они были потеряны в каналах), затем записать и послать изменение координатору.
- Координатор рассылает всем (кроме автора) модификацию.
- Приняв модификацию, процесс проверяет, получил ли он предыдущие модификации, если не получил, то запрашивает их. Применяет все полученные модификации.

б)при чтении:
Производится обращение к локальной копии данных.

в)значения модифицированных переменных рассылаются координатору после записи(см п (а))
а также координатором автору новых модификаций после получения от него запроса на запись.

г) блокируется ли процесс на время выполнения записи или рассылки значений переменных:
процесс – автор блокируется до записи для получения консистентной версии памяти.

Считаем, что всего 10 процессов/процессоров. 1 из них – координатор.
10 процессов модифицируют 10 переменных.
Каждый процесс посылает координатору запрос на модификацию. 9 сообщений (1 сообщение будет локальным). В ответ на каждый запрос на модификацию координатор вышлет 9 номеров и 9 модификаций. Координатору будут переданы 9 модифицированных блоков данных.
Итого получается 9*(9+9) сообщений = 9*(18* Ts+9*Tb*Ln+9*Tb*Lm+5*Tb*Lm)

2. **Причинная консистентность памяти и алгоритм ее реализации в DSM с полным размножением при условии, что никаких сведений от компилятора о причинной зависимости операций записи не имеется. Сколько времени потребует модификация 10 различных переменных, если все 10 процессов (каждый процесс модифицирует одну переменную), находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на модификацию своей переменной. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

При реализации причинной консистентности для случая размножения страниц выполнение операций с общей памятью требует ожидания выполнения только тех предыдущих операций записи, которые являются потенциально причинно зависимыми. Параллельные операции записи не задерживают выполнение операций с общей памятью, а также не требуют неделимости широковещательных рассылок.
Определение потенциальной причинной зависимости может осуществляться компилятором посредством анализа зависимости операторов программы по данным.
Система DSM может это осуществить посредством нумерации всех записей на каждом процессоре, распространения этих номеров по всем процессорам вместе с модифицируемыми данными, и задержке любой модификации на любом процессоре до тех пор, пока он не получит все те модификации, о которых известно процессору - автору задерживаемой модификации.

3. **Процессорная консистентность памяти и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует модификация 10 различных переменных, если все 10 процессов (каждый процесс модифицирует одну переменную), находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на модификацию своей переменной. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Процессорная консистентность – каждый процессор видит модификации, производимые одним процессором, в том же порядке, как они были произведены. Кроме того по каждой переменной есть согласие относительно поядка её модификаций.

Таким образом процессор, меняя переменную, должен посылать результаты координатору и дожидаться консистентного состояния, когда к нем придут все изменения, которые произошли до его модификации включая его собственную.

Координатор хранит последовательность модификаций. Начало последовательности выбирается так, чтобы изменения, отсутствующие у какого-то процесса были в хранимой последовательности.

а) при записи
Процесс посылает координатору свои модификации, после чего координатор высылает процессу имеющийся набор модификаций, относящихся к тому-же блоку данных.
б) при чтении
Процесс берёт данные из своей локальной копии.
в) значения модифицируемых переменных рассылается
координатору при изменении
процессу, если они относятся к его данным
процессам периодически (по таймеру/ при определенной длины последовательности и т.д) --- это приводит все копии данных в одинаковое состояние.

г) процесс блокируется на время выполнения записи до получения актуальных модификаций от координатора.

Временная оценка: в лучшем случае модификация закончится за 20*(Ts+ Tb*Lm). При этом ещё за время 10*(Ts+Tb*9*Lm) все процессы получат полный набор модификаций.

4. **PRAM консистентность памяти и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует 3-кратная модификация 10 различных переменных, если все 10 процессов (каждый процесс 3 раза модифицирует одну переменную), находящихся на разных ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания), одновременно выдали запрос на модификацию. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

PRAM консистентность – действия каждого процесса правильно упорядочены с т.зрения любого другого процесса. Взгляд на последовательность действий разных процессов может отличаться для двух процессов.

Производя запись процессор должен записать в локальную память и разослать всем свои изменения.Здесь требуется надёжность при передаче сообщений по каналам при сохранении их порядка.

а)при записи - рассылаются изменения всем через координатора.
б)при чтении - данные берутся из локальной памяти
в) значения модифицируемых переменных рассылаются другим процессам при изменении(записи) переменных.
г) процесс на время выполнения записи не блокируется

Потребуется 9*3* (Ts+Tb*Lm)

5. **<u>Слабая консистентность памяти</u> и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует модификация одним процессом 10 обычных переменных, а затем 3-х различных синхронизационных переменных, если DSM реализована на 10 ЭВМ сети с шинной организацией (<u>с аппаратными возможностями широковещания</u>). Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Слабая консистентность:– доступ к синхронизационным переменным – по последовательной консистентности. – доступ к синхронизационным переменным блокируется, пока не выполнились все операции записи других процессов, обращавшихся к синхр. переменной. – доступ к данным запрещен пока не выполнились обращения к синзрон. перем.

а) при модификации обычных данных записываются в локальную память.
при модификации синхронизационных переменных все модификации данных посылаются координатору, который номерует модификации, рассылает модификации широковещательно и возвращает посылающему номер последней принятой модификации. Если отправитель не имеет какой-либо записи, он должен её потребовать.

б) при чтении обычных данных они берутся из локальной памяти
при чтении синхр. переменных обращение к координатору происходит так же, как при записи.

в) значения модифицируемых переменных рассылаются координатору после обращения к синхр. перем. и далее координатором либо при обработке такого обращения, либо когда требуют пропущенные фрагменты модификаций.

г) процесс блокируется при обращении к синхр. переменным;

Временная оценка: Ts+Tb*Lm*11+Ts*2+Tb*Lm*2= 3*Ts+13*Lm

6. **<u>Консистентность памяти по выходу</u> и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует трехкратное выполнение каждым процессом критической секции, в которой модифицируются 10 переменных, если DSM реализована на 10 ЭВМ сети с шинной организацией (<u>с аппаратными возможностями широковещания</u>). Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

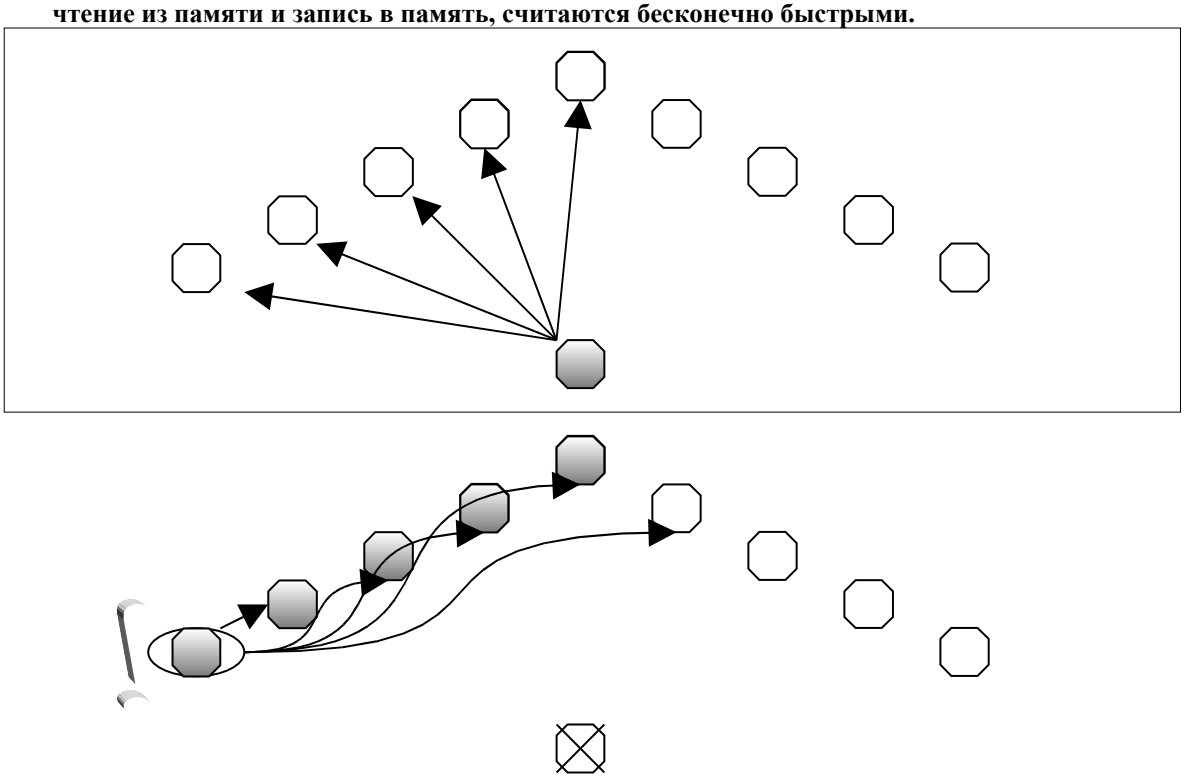
3*10*(2*(Ts+Tb*Lcs)+(Ts+Tb*Lv*10+Tb*Lcs))

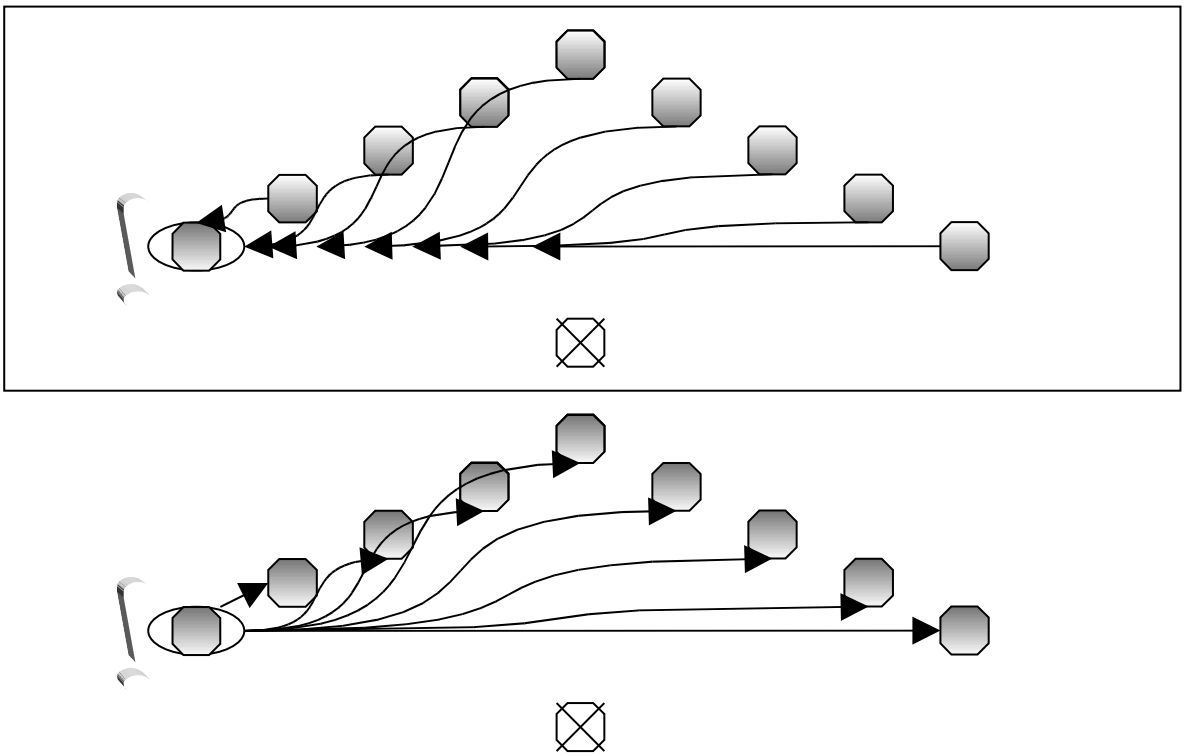
7. **<u>Консистентность памяти по входу</u> и алгоритм ее реализации в DSM с полным размножением. Сколько времени потребует трехкратное выполнение критической секции и модификация в ней 10 переменных каждым процессом, если DSM реализована на 10 ЭВМ сети с шинной организацией (с аппаратными возможностями широковещания). Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память считаются бесконечно быстрыми.**

3*10*(2*(Ts+Tb*Lcs)+(Ts+Tb*Lv*10+Tb*Lcs))

Тема-7

1. **Алгоритм надежных и неделимых широковещательных рассылок сообщений. Дайте оценку времени выполнения одной операции рассылки для сети из 10 ЭВМ с шинной организацией (без аппаратных возможностей широковещания), если отправитель сломался после посылки 5-го сообщения. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**





Итого получаются следующие результаты: 5+8 отправок сообщения, 5+8 запросов, 8 уведомлений, 5+8 ответов.
Времени надо: 13*(Ts+N*Tb)+13*(Ts+Lz*Tb)+8*(Ts+Ln*Tb)+13(Ts+La*Tb)
Выделив основные компоненты, получим: 47*Ts+13*N*Tb

2. **Протоколы голосования. Алгоритмы и применение. Дайте оценку времени выполнения одним процессом 2-х операций записи и 10 операций чтения N байтов информации с файлом, расположенным (размноженным) на остальных 10 ЭВМ сети с шинной организацией (без аппаратных возможностей широковещания). Определите оптимальные значения кворума чтения и кворума записи для N=300. Время старта (время «разгона» после получения доступа к шине для передачи) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса (при одновременных запросах - в порядке номеров ЭВМ). Операции с файлами и процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Хотя собственно алгоритм предписывает назначить каждой копии отдельное число голосов, будем считать, что у каждой копии ровно 1 голос.

Одна запись в память потребует:

Vw*(Ts+Tb*Lz)+Vw*(Ts+Tb*Lo)+Vw*(Ts+Tb*N), или приведя Vw

Vw*(3*Ts+Tb*Lz+Tb*Lo+Tb*N)

Возможно, если писатель не обладает актуальной копией, ему придется запросить её за

Ts+Tb*Lz+Ts+Tb*N

Одно чтение потребует:

Vr*(Ts+Tb*Lz)+Vr*(Ts+Tb*Lo)

Возможно, если читатель не обладает актуальной копией, ему придется запросить её за

Ts+Tb*Lz+Ts+Tb*N

Получается, что надо минимизировать Vw(1000+2Lz+2Lo)+Vr(2000+10Lz+10Lo)
Наилучший результат будет при Vw=9, Vr=1. То есть, согласие на запись должны дать все, согласие на чтение должен дать любой процесс.

3. **Консистентное и строго консистентные множества контрольных точек. Дайте оценку накладных расходов на синхронную фиксацию строго консистентного множества контрольных точек для сети из 10 ЭВМ с шинной организацией (без аппаратных возможностей широковещания), если накладные расходы на синхронную фиксацию консистентного множества равны T1. Время старта (время «разгона» после получения доступа к шине для передачи сообщения) равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Доступ к шине ЭВМ получают последовательно в порядке выдачи запроса на передачу (при одновременных запросах - в порядке номеров ЭВМ). Процессорные операции, включая чтение из памяти и запись в память, считаются бесконечно быстрыми.**

Шаг 1. Рассылка всем сообщений с требованием начать фиксацию локальной контрольной точки. Процессы должны перестать посылать сигналы (кроме служебных), но обязательно принимать приходящие по линиям связи сигналы (и служебные и не служебные). Процессы должны запоминать сигналы в своих буферах.
Шаг 2. Получение результатов операции.
Шаг 3. Когда 2 успешно, то есть все процессы перестали посылать сигналы.
Посылается операция проверки каналов:

Каждый процесс должен получив такое сообщение поставить его в очередь для данного канала.

Когда сообщение пройдет через данный канал, значит все неслужебные сообщения были отправлены.

Если сообщение проверки приходит по прочищенному каналу или в очереди канала уже стоит такое сообщение, тогда не следует его распространять дальше.

Шаг 4: Когда 3 успешно, то есть все процессоры освободили свои исходящие очереди. Когда у каждого процессора все входящие каналы прочищены (по ним получен сигнал очистки), тогда он должен послать сигнал готовности к фиксации строго консистентной контрольной точки.
Шаг 5: Когда координатор получит все сообщения о готовности к строго консистентной контрольной точке, он принимает её.

Если известно, что физическая среда единая и очередь сообщений к ней одна, тогда координатор делает следующее: рассылает всем сообщение о прочистке канала. Каждый узел должен вернуть это сообщение. Получив любое информационное сообщение процесс его буферизует и фиксирует новую точку консистентности. Получив все сообщения координатор вправе считать, что все процессы строго консистентны. После чего он должен разослать сообщение о принятии строго консистентного множества контрольных точек. При получении такого сообщения процесс обязан зафиксировать имеющуюся контрольную точку как принадлежащую множеству строго консистентных КТ и дальше её не модифицировать.
Далее каждый процесс должен запросить разрешение на посылку сообщений (на нормальный режим работы). Собрав ВСЕ разрешения координатор знает, что строго консистентное множество КТ зафиксировано и удовлетворяет запросы, переводя систему в нормальный режим.
Сложность: 9+9+9+9+9+9+9= 9*7 = 63*(Ts+L*Tb) , L мало

Замечания.
При решении задач по теме 2 обратить внимание на следующее.

1. Нельзя модифицировать общие переменные вне КС.
2. При наличии вложенных КС или запросов нескольких семафоров необходимо убедиться, что не могут возникнуть тупики.
3. Нельзя обращаться к семафорам и событиям обычными операторами – только посредством операций, которые определены над ними (P, V, POST, WAIT, CLEAR).
4. Нельзя освобождать свободный семафор и объявлять уже объявленное событие.
5. Определять начальные значения семафоров и событий, если они должны быть отличны от нуля (семафор занят, событие не объявлено).

При ответах на вопросы по теме 6 следовать следующему плану.

1. Определение модели консистентности.
2. Алгоритм реализации в DSM с полным размножением (много писателей и много читателей, каждый из которых имеет свою копию всех переменных). Алгоритм должен быть корректным для любой коммуникационной сети и обеспечивать высокую эффективность для конкретной сети, указанной в задаче. Описание алгоритма должно содержать ответы на следующие вопросы:
   а) что делается при записи;
   б) что делается при чтении;
   в) когда, кому и как рассылаются значения модифицируемых переменных;
   г) блокируется ли процесс на время выполнения записи или рассылки значений переменных;
   д) если речь идет о моделях консистентности, связанных с синхронизацией, то указать алгоритм синхронизации (например, алгоритм входа в КС и выхода из нее).
3. Оценить время работы описанного в пункте 2 алгоритма применительно к конкретной задаче.