

**Національний технічний університет України
“Київський політехнічний інститут ім. Ігоря
Сікорського”**

**Факультет прикладної математики
Кафедра системного програмування і спеціалізованих
комп’ютерних систем**

Лабараторна робота №4
з дисциплін
«Паралельне програмування»

**ТЕМА: «КОМПЛЕКСНЕ ВИКОРИСТАННЯ ЗАСОБІВ ВЗАЄМОДІЇ
ПАРАЛЕЛЬНИХ ПОТОКІВ ОПЕРАЦІЙНОЇ СИСТЕМИ LINUX»**

Виконала: Поліщук М.І.
Студентка групи: КВ-33
Перевірив: Марченко О.І.
Оцінка:

Постановка завдання

1. Опрацювати всі надані лектором приклади коду паралельних потоків по темі «Засоби взаємодії паралельних потоків операційної системи Linux», що знаходяться в директоріях **04_Common_Resource** та **05_Atomic_Operations**, тобто:
 - вміти запускати всі ці приклади і отримувати результати на захисті лабораторної роботи;
 - знати які структури даних та конструкції взаємодії паралельних потоків описані в коді кожного прикладу та як вони працюють, а також вміти це пояснити на захисті лабораторної роботи;
 - розібратися з теоретичними ситуаціями, які відображують дані приклади, а також вміти їх розказати та пояснити на захисті лабораторної роботи;
 - бути готовими до виконання модифікацій будь-яких з цих прикладів на захисті лабораторної роботи.
2. Написати програму, яка реалізує роботу паралельних потоків згідно заданої за варіантом схеми. Особливості реалізації синхронізації паралельних потоків та взаємного виключення потоків при доступі до спільних ресурсів задані за варіантами у таблицях 1 та 2.
3. При написанні програми виконати повне трасування роботи програми за допомогою операторів друку, тобто розставити в програмі оператори друку таким чином, щоб можна було прослідкувати всі варіанти виконання паралельних потоків і впевнитись у коректності роботи програми. Протокол трасування рекомендується записувати у файл (log-файл).
4. Запуск усіх потоків повинен бути виконаний у головній програмі.
5. Кожен потік повинен бути організованим у вигляді нескінченного циклу.
6. Всі дії задані за варіантами, що вказані у таблиці, повинні бути виконані всередині цього нескінченного циклу.
7. Взаємне розташування операторів синхронізації та доступу до спільного ресурсу, якщо вони знаходяться у одному потоці, є довільним.

8. Оскільки синхронізація за допомогою семафорів SCR21, SCR22 згідно завдання розташована всередині нескінченних циклів, то відразу після виконання синхронізації ці семафори повинні бути знову встановлені у початковий закритий стан.
9. Закінчення програми можна виконати двома способами:
- примусовим перериванням за допомогою натиснення комбінації клавіш Ctrl+C;
 - оператором break при виконанні умови, яка стає істинною, коли буфер спільного ресурсу повністю заповнюється і повністю звільняється мінімум по два рази.
10. Якщо при реалізації паралельних потоків була використана функція usleep(), то передбачити режим запуску програми з «відключеними» функціями usleep().
11. Виконати налагодження написаної програми.

Завдання за варіантом

Варіант 13

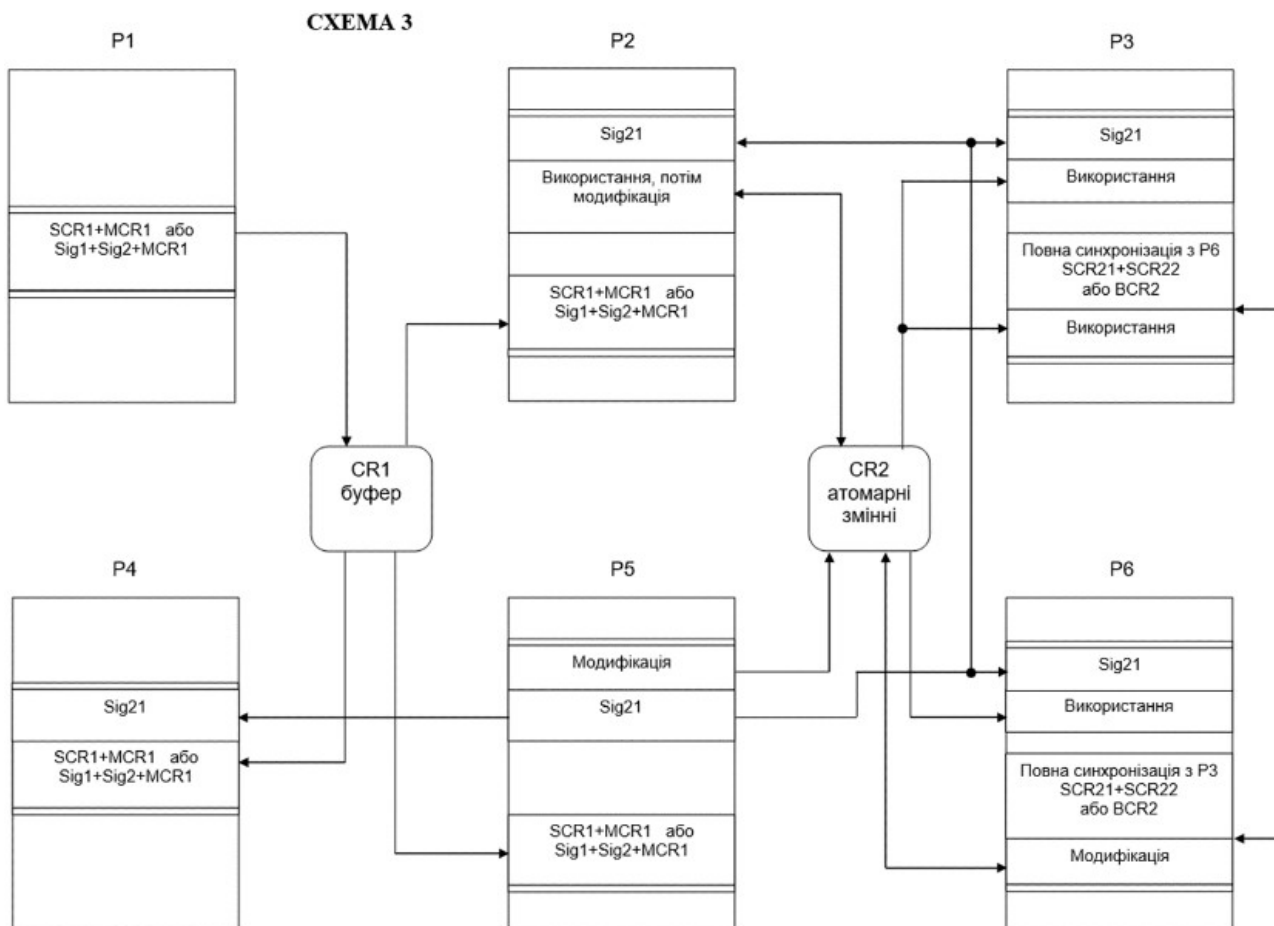
Таблиця 1

№ варіанту	№ схеми	Спільний ресурс 1 CR1 (буфер обміну даними)		Спільний ресурс 2 CR2	Засоби синхронізації паралельних потоків				
					Семафори та бар'єр для повної або неповної синхронізації потоків			Сигнальні (умовні) змінні синхронізації потоків	
		Структура даних, що використовується у якості спільного ресурсу 1 (CR1)	Засоби взаємного виключення при доступі до спільного ресурсу 1 SCR1 + MCR1 або Sig1+Sig2+MCR1	Атомарні дані (взяти по 2 змінних кожного з типів: <i>int</i> , <i>unsigned</i> , <i>long</i> , <i>long unsigned</i>) та операції (табл. 2)	Вид двійкового семафору SCR21	Вид двійкового семафору SCR22	Бар'єр BCR2	Вид сигналу сигнальної (умовної) змінної Sig21	Вид сигналу сигнальної (умовної) змінної Sig22
13	3	Цикл.буфер (ДСД)	Неблокуючий багатозначний семафор SCR1 та неблокуючий м'ютекс MCR1	2, 7, 3, 4, 11, 12, 13, 14	—	—	BCR2	Багато-значний	—

Таблиця 2 містить перелік вбудованих функцій, які реалізують атомарні операції.

Таблиця 2

Номер атомарної операції (функції)	Ідентифікатор вбудованої функції, що відповідає атомарній операції
1.	<i>type __atomic_add_fetch (type *ptr, type val, int memorder)</i>
2.	<i>type __atomic_sub_fetch (type *ptr, type val, int memorder)</i>
3.	<i>type __atomic_and_fetch (type *ptr, type val, int memorder)</i>
4.	<i>type __atomic_xor_fetch (type *ptr, type val, int memorder)</i>
5.	<i>type __atomic_or_fetch (type *ptr, type val, int memorder)</i>
6.	<i>type __atomic_nand_fetch (type *ptr, type val, int memorder)</i>
7.	<i>type __atomic_fetch_add (type *ptr, type val, int memorder)</i>
8.	<i>type __atomic_fetch_sub (type *ptr, type val, int memorder)</i>
9.	<i>type __atomic_fetch_and (type *ptr, type val, int memorder)</i>
10.	<i>type __atomic_fetch_xor (type *ptr, type val, int memorder)</i>
11.	<i>type __atomic_fetch_or (type *ptr, type val, int memorder)</i>
12.	<i>type __atomic_fetch_nand (type *ptr, type val, int memorder)</i>
13.	<i>bool __atomic_compare_exchange_n (type *ptr, type *expected, type desired, bool weak, int success_memorder, int failure_memorder)</i>
14.	<i>void __atomic_exchange (type *ptr, type *val, type *ret, int memorder)</i>



Код програми:

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <unistd.h>
#include <time.h>

#define USLEEP_TIME 800
#define N 16
#define FULL_SYNC_NUMBER 2

pthread_t P1, P2, P3, P4, P5, P6;

//засоби взаємного виключення
pthread_mutex_t mcr1 = PTHREAD_MUTEX_INITIALIZER;
sem_t scr1;
int usleep_flag = 0; //Flag to toggle usleep in Consumers threads (p4, p2, p5)

//засоби синхронізації паралельних потоків
pthread_barrier_t bcr2;
pthread_cond_t Sig21 = PTHREAD_COND_INITIALIZER;
pthread_mutex_t mutex_Sig21 = PTHREAD_MUTEX_INITIALIZER;
int Sig21_p2_flag = 0;
int Sig21_p3_flag = 0;
int Sig21_p4_flag = 0;
int Sig21_p6_flag = 0;

//атомарні змінні
int ivar1 = 1, ivar2 = 2;
unsigned uvar1 = 3, uvar2 = 4;
long lvar1 = 5, lvar2 = 6;
long unsigned luvar1 = 7, luvar2 = 8;

void use_Atomic(int thread_number);
void modify_Atomic(int thread_number);

//Структура спільного ресурсу та необхідні допоміжні засоби
int max_list_size = N;
int elementCounter = 0;
int total_element_created = 0;

struct node {
    int data;
    struct node *next;
};

struct node *tail = NULL;

void addToEmpty(int value) {
    struct node *temp = malloc(sizeof(struct node));
    temp->data = value;
    temp->next = temp;
    tail = temp;
}

void addAtEnd(int value) {
    if (tail == NULL) {
        addToEmpty(value);
        return;
    }

    struct node *temp = malloc(sizeof(struct node));
```

```

temp->data = value;
temp->next = tail->next;

tail->next = temp;
tail = tail->next;
}

int delFirst(void) {
    int ret_data;

    if(tail == NULL)
        return -1;

    if(tail->next == tail) {
        ret_data = tail->data;
        free(tail);
        tail = NULL;
        return ret_data;
    }

    struct node *temp;
    temp = tail->next;
    ret_data = temp->data;
    tail->next = temp->next;
    free(temp);
    temp = NULL;
    return ret_data;
}

void freeBuffer(void) {
    if(tail == NULL)
        return;

    if(tail->next == tail) {
        free(tail);
        tail = NULL;
        return;
    }

    struct node *temp = NULL;
    while(tail->next != tail) {
        temp = tail->next;
        tail->next = temp->next;
        free(temp);
    }

    free(tail);
    tail = NULL;
    return;
}

void print_CR1(void) {
    if(tail == NULL) {
        printf("List is empty\n");
    }
    else {
        printf("Content of the Circular Linked List: ");
        struct node *temp = tail->next;
        do {
            printf("%d ", temp->data);
            temp = temp->next;
        } while(temp != tail->next);
        printf("\n");
    }
}

```

```
}  
}
```

```
void modify_Atomic(int thread_number)  
{  
    printf("\nP%d modifies atomic variables using numbered atomic ops\n",  
thread_number);  
  
    // 2. __atomic_add_fetch  
    printf("(2) int add fetch: %d\n", __atomic_add_fetch(&ivar1, ivar2,  
__ATOMIC_RELAXED));  
  
    // 8. __atomic_fetch_sub  
    printf("(8) int fetch sub: %d\n", __atomic_fetch_sub(&ivar2, ivar1,  
__ATOMIC_RELAXED));  
  
    // 4. __atomic_xor_fetch  
    printf("(4) long xor fetch: %ld\n", __atomic_xor_fetch(&lvar1, lvar2,  
__ATOMIC_RELAXED));  
  
    // 5. __atomic_or_fetch  
    printf("(5) long or fetch: %ld\n", __atomic_or_fetch(&lvar2, lvar1,  
__ATOMIC_RELAXED));  
  
    // 10. __atomic_fetch_xor  
    printf("(10) unsigned fetch xor: %u\n", __atomic_fetch_xor(&uvar1, uvar2,  
__ATOMIC_RELAXED));  
  
    // 11. __atomic_fetch_or  
    printf("(11) unsigned fetch or: %u\n", __atomic_fetch_or(&uvar2, uvar1,  
__ATOMIC_RELAXED));  
  
    // 13. __atomic_compare_exchange_n  
    printf("(13) long unsigned compare exchange success: %d\n",  
__atomic_compare_exchange_n(&luvar1, &luvar2, thread_number, 0,  
__ATOMIC_RELAXED, __ATOMIC_RELAXED));  
  
    // 14. __atomic_exchange  
    printf("(14) long unsigned before exchange: %lu %lu\n", luvar1, luvar2);  
    __atomic_exchange(&luvar2, &luvar1, &luvar1, __ATOMIC_RELAXED);  
    printf("(14) long unsigned after exchange: %lu %lu\n", luvar1, luvar2);  
}  
  
void use_Atomic(int thread_number)  
{  
    printf("\nP%d uses atomic variables\n", thread_number);  
    printf("ivar1 + ivar2 = %d + %d = %d\n", ivar1, ivar2, (ivar1 + ivar2));  
    printf("uvar1 + uvar2 = %u + %u = %u\n", uvar1, uvar2, (uvar1 + uvar2));  
    printf("lvar1 + lvar2 = %ld + %ld = %ld\n", lvar1, lvar2, (lvar1 + lvar2));  
    printf("luvar1 + luvar2 = %lu + %lu = %lu\n\n", luvar1, luvar2, (luvar1 +  
luvar2));  
}
```

```
void *P1_thread(void *arg) {  
    printf("Thread P1 has started\n");  
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);  
  
    int sem_value = 0;  
    while(1) {  
        sem_getvalue(&scr1, &sem_value);  
        if(sem_value < max_list_size) {
```

```

        printf("P1 Producer tries to lock MCR1\n");
        while(pthread_mutex_trylock(&mcr1) != 0) {
            //P1 Producer doing something useful while mutex mcr1 is locked
            by other thread
        }
        printf("P1 Producer successfully locked MCR1\n");

        addAtEnd(total_element_created);
        total_element_created++;
        printf("P1 Producer successfully added a new element number %d to
CR1 list\n", elementCounter);
        elementCounter++;
        //print_CR1();
        sem_post(&scr1);
        printf("P1 Producer sent SCR1 post signal\n");
        pthread_mutex_unlock(&mcr1);
        printf("P1 Producer unlocked MCR1\n");

    }

}
return NULL;
}

```

```

void *P2_thread(void *arg) {
    printf("Thread P2 has started\n");
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
    int data;

    while(1) {

        printf("P2 tries to lock Sig21\n");
        pthread_mutex_lock(&mutex_Sig21);
        printf("P2 Consumer waits for Sig21\n");
        while(!Sig21_p2_flag) {
            pthread_cond_wait(&Sig21, &mutex_Sig21);
        }
        printf("P2 Consumer got Sig21\n");

        Sig21_p2_flag = 0;
        pthread_mutex_unlock(&mutex_Sig21);
        printf("P2 unlocked Sig21\n");

        use_Atomic(2);
        modify_Atomic(2);

        printf("P2 Consumer waits for SCR1 post signal\n");
        while(sem_trywait(&scr1) != 0) {
            //P2 Consumer doing something useful while there is no elements
        }
        printf("P2 Consumer got SCR1 post signal\n");

        printf("P2 Consumer tries to lock MCR1\n");
        while(pthread_mutex_trylock(&mcr1) != 0) {
            //P2 Consumer doing something useful while mutex mcr1 is locked by
            other thread
        }
        printf("P2 Consumer successfully locked MCR1\n");

        data = delFirst();
        elementCounter--;
        printf("P2 Consumer successfully took %d from CR1 list\n", data);
        printf("Now there is %d elements left in CR1 list\n", elementCounter);
    }
}

```



```

        pthread_mutex_unlock(&mcr1);
        printf("P2 Consumer unlocked MCR1\n");

        if(usleep_flag)
            usleep(USLEEP_TIME);
    }
    return NULL;
}

void *P3_thread(void *arg) {
    printf("Thread P3 has started\n");
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);

    while(1) {
        printf("P3 tries to lock Sig21\n");
        pthread_mutex_lock(&mutex_Sig21);
        printf("P3 waits for Sig21\n");
        while(!Sig21_p3_flag) {
            pthread_cond_wait(&Sig21, &mutex_Sig21);
        }
        printf("P3 got Sig21\n");

        Sig21_p3_flag = 0;
        pthread_mutex_unlock(&mutex_Sig21);
        printf("P3 unlocked Sig21\n");

        use_Atomic(3);

        printf("P3 waits on Barrier bcr2\n");
        pthread_barrier_wait(&bcr2);
        printf("P3 passed through Barrier bcr2\n");

        use_Atomic(3);
    }
    return NULL;
}

void *P4_thread(void *arg) {
    printf("Thread P4 has started\n");
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
    int data;

    while(1) {

        printf("P4 tries to lock Sig21\n");
        pthread_mutex_lock(&mutex_Sig21);
        printf("P4 Consumer waits for Sig21 \n");
        while(!Sig21_p4_flag) {
            pthread_cond_wait(&Sig21, &mutex_Sig21);
        }
        printf("P4 Consumer got Sig21\n");

        Sig21_p4_flag = 0;
        pthread_mutex_unlock(&mutex_Sig21);
        printf("P4 Consumer unlocked Sig21\n");

        printf("P4 Consumer waits for SCR1 post signal\n");
        while(sem_trywait(&scr1) != 0) {
            //P4 Consumer doing something useful while there is no elements
        }
        printf("P4 Consumer got SCR1 post signal\n");
    }
}

```

```

        printf("P4 Consumer tries to lock MCR1\n");
        while(pthread_mutex_trylock(&mcr1) != 0) {
            //P4 Consumer doing something useful while mutex mcr1 is locked by
other thread
        }
        printf("P4 Consumer successfully locked MCR1\n");

        data = delFirst();
        elementCounter--;
        printf("P4 Consumer successfully took %d from CR1 list\n", data);
        printf("Now there is %d elements left in CR1 list\n", elementCounter);
        pthread_mutex_unlock(&mcr1);
        printf("P4 Consumer unlocked MCR1\n");

        if(usleep_flag)
            usleep(USLEEP_TIME);
    }
    return NULL;
}

```

```

void *P5_thread(void *arg) {
    printf("Thread P5 has started\n");
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);
    int data;

    while(1) {
        modify_Atomic(5);

        printf("P5 Consumer tries to lock Sig21\n");
        pthread_mutex_lock(&mutex_Sig21);
        Sig21_p2_flag = 1;
        Sig21_p3_flag = 1;
        Sig21_p4_flag = 1;
        Sig21_p6_flag = 1;

        pthread_cond_broadcast(&Sig21);
        printf("P5 Consumer broadcasted Sig21\n");

        pthread_mutex_unlock(&mutex_Sig21);
        printf("P5 Consumer unlocked Sig21\n");

        printf("P5 Consumer waits for SCR1 post signal\n");
        while(sem_trywait(&scr1) != 0) {
            //P5 Consumer doing something useful while there is no elements
        }
        printf("P5 Consumer got SCR1 post signal\n");

        printf("P5 Consumer tries to lock MCR1\n");
        while(pthread_mutex_trylock(&mcr1) != 0) {
            //P5 Consumer doing something useful while mutex mcr1 is locked by
other thread
        }
        printf("P5 Consumer successfully locked MCR1\n");

        data = delFirst();
        elementCounter--;
        printf("P5 Consumer successfully took %d from CR1 list\n", data);
        printf("Now there is %d elements left in CR1 list\n", elementCounter);
        pthread_mutex_unlock(&mcr1);
        printf("P5 Consumer unlocked MCR1\n");

        if(usleep_flag)
            usleep(USLEEP_TIME);
    }
}

```

```

    }
    return NULL;
}

void *P6_thread(void *arg) {
    printf("Thread P6 has started\n");
    pthread_setcanceltype (PTHREAD_CANCEL_ASYNCHRONOUS, NULL);

    while(1) {

        printf("P6 tries to lock Sig21\n");
        pthread_mutex_lock(&mutex_Sig21);

        printf("P6 waits for Sig21\n");
        while(!Sig21_p6_flag) {
            pthread_cond_wait(&Sig21, &mutex_Sig21);
        }
        printf("P6 got signal Sig21\n");

        Sig21_p6_flag = 0;
        pthread_mutex_unlock(&mutex_Sig21);
        printf("P6 unlocked Sig21\n");

        use_Atomic(6);

        printf("P6 waits on Barrier bcr2\n");
        pthread_barrier_wait(&bcr2);
        printf("P6 passed through Barrier bcr2\n");

        modify_Atomic(6);
    }
    return NULL;
}

int main(void) {

    pthread_barrier_init(&bcr2, NULL, FULL_SYNC_NUMBER);
    sem_init(&scr1, 0, 0);

    pthread_create(&P1, NULL, P1_thread, NULL);
    pthread_create(&P2, NULL, P2_thread, NULL);
    pthread_create(&P3, NULL, P3_thread, NULL);
    pthread_create(&P4, NULL, P4_thread, NULL);
    pthread_create(&P5, NULL, P5_thread, NULL);
    pthread_create(&P6, NULL, P6_thread, NULL);

    pthread_join(P1, NULL);
    pthread_join(P2, NULL);
    pthread_join(P3, NULL);
    pthread_join(P4, NULL);
    pthread_join(P5, NULL);
    pthread_join(P6, NULL);

    printf("Finished!!!\n");

    pthread_barrier_destroy(&bcr2);
    sem_destroy(&scr1);
    freeBuffer();

    return 0;
}

```

Протокол роботи програми:

Thread P1 has started

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P1 Producer unlocked MCR1

Thread P5 has started

P5 modifies atomic variables using numbered atomic ops

(2) int add fetch: 3

(8) int fetch sub: 2

(4) long xor fetch: 3

(5) long or fetch: 7

(10) unsigned fetch xor: 3

(11) unsigned fetch or: 4

(13) long unsigned compare exchange success: 0

(14) long unsigned before exchange: 7 7

(14) long unsigned after exchange: 7 7

P5 Consumer tries to lock Sig21

P5 Consumer broadcasted Sig21

P5 Consumer unlocked Sig21

P5 Consumer waits for SCR1 post signal

Thread P6 has started

P6 tries to lock Sig21

P6 waits for Sig21

P6 got signal Sig21

P6 unlocked Sig21

Thread P2 has started

P2 tries to lock Sig21

P2 Consumer waits for Sig21

P2 Consumer got Sig21

P2 unlocked Sig21

P2 uses atomic variables

$\text{ivar1} + \text{ivar2} = 3 + -1 = 2$

$\text{uvar1} + \text{uvar2} = 7 + 7 = 14$

$\text{lvar1} + \text{lvar2} = 3 + 7 = 10$

$\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P2 modifies atomic variables using numbered atomic ops

(2) int add fetch: 2

(8) int fetch sub: -1

(4) long xor fetch: 4

(5) long or fetch: 7

(10) unsigned fetch xor: 7

(11) unsigned fetch or: 7

(13) long unsigned compare exchange success: 1

(14) long unsigned before exchange: 2 7

(14) long unsigned after exchange: 7 2

P2 Consumer waits for SCR1 post signal
Thread P4 has started
P4 tries to lock Sig21
P4 Consumer waits for Sig21
P4 Consumer got Sig21
P4 Consumer unlocked Sig21
P4 Consumer waits for SCR1 post signal
Thread P3 has started
P3 tries to lock Sig21
P3 waits for Sig21
P3 got Sig21
P3 unlocked Sig21

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 2 + -3 = -1$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 2 = 9$

P3 waits on Barrier bcr2
P5 Consumer got SCR1 post signal
P5 Consumer tries to lock MCR1
P5 Consumer successfully locked MCR1

P6 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 2 + -3 = -1$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 2 = 9$

P6 waits on Barrier bcr2
P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops
(2) int add fetch: -1
(8) int fetch sub: -3
P3 passed through Barrier bcr2

P3 uses atomic variables
(4) long xor fetch: 3
P5 Consumer successfully took 0 from CR1 list
 $\text{ivar1} + \text{ivar2} = -1 + -2 = -3$
(5) long or fetch: 7
Now there is 0 elements left in CR1 list
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 2 = 9$

(10) unsigned fetch xor: 0
(11) unsigned fetch or: 7
P5 Consumer unlocked MCR1
(13) long unsigned compare exchange success: 0

(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P2 Consumer got SCR1 post signal
P2 Consumer tries to lock MCR1
P2 Consumer successfully locked MCR1
P1 Producer unlocked MCR1
P2 Consumer successfully took 1 from CR1 list
Now there is 0 elements left in CR1 list
P2 Consumer unlocked MCR1

P5 modifies atomic variables using numbered atomic ops

(2) int add fetch: -3
(8) int fetch sub: -2
(4) long xor fetch: 4
(5) long or fetch: 7
(10) unsigned fetch xor: 7
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 1
(14) long unsigned before exchange: 5 7
(14) long unsigned after exchange: 7 5
P5 Consumer tries to lock Sig21
P5 Consumer broadcasted Sig21
P5 Consumer unlocked Sig21
P5 Consumer waits for SCR1 post signal
P6 tries to lock Sig21
P6 waits for Sig21
P6 got signal Sig21
P6 unlocked Sig21

P6 uses atomic variables

$\text{ivar1} + \text{ivar2} = -3 + 1 = -2$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 5 = 12$

P6 waits on Barrier bcr2

P3 tries to lock Sig21
P3 waits for Sig21
P3 got Sig21
P3 unlocked Sig21

P3 uses atomic variables

$\text{ivar1} + \text{ivar2} = -3 + 1 = -2$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 5 = 12$

P3 waits on Barrier bcr2

P3 passed through Barrier bcr2

P3 uses atomic variables

$\text{ivar1} + \text{ivar2} = -3 + 1 = -2$

$\text{uvar1} + \text{uvar2} = 0 + 7 = 7$

$\text{lvar1} + \text{lvar2} = 4 + 7 = 11$

$\text{luvar1} + \text{luvar2} = 7 + 5 = 12$

P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops

(2) int add fetch: -2

(8) int fetch sub: 1

(4) long xor fetch: 3

(5) long or fetch: 7

(10) unsigned fetch xor: 0

(11) unsigned fetch or: 7

(13) long unsigned compare exchange success: 0

(14) long unsigned before exchange: 7 7

(14) long unsigned after exchange: 7 7

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P1 Producer unlocked MCR1

P5 Consumer got SCR1 post signal

P5 Consumer tries to lock MCR1

P5 Consumer successfully locked MCR1

P5 Consumer successfully took 2 from CR1 list

Now there is 0 elements left in CR1 list

P5 Consumer unlocked MCR1

P2 tries to lock Sig21

P2 Consumer waits for Sig21

P2 Consumer got Sig21

P2 unlocked Sig21

P2 uses atomic variables

$\text{ivar1} + \text{ivar2} = -2 + 3 = 1$

$\text{uvar1} + \text{uvar2} = 7 + 7 = 14$

$\text{lvar1} + \text{lvar2} = 3 + 7 = 10$

$\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P2 modifies atomic variables using numbered atomic ops

(2) int add fetch: 1

(8) int fetch sub: 3

(4) long xor fetch: 4

(5) long or fetch: 7

(10) unsigned fetch xor: 7

(11) unsigned fetch or: 7

(13) long unsigned compare exchange success: 1

(14) long unsigned before exchange: 2 7

(14) long unsigned after exchange: 7 2
P2 Consumer waits for SCR1 post signal
P6 tries to lock Sig21
P6 waits for Sig21
P3 tries to lock Sig21
P3 waits for Sig21
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1

P5 modifies atomic variables using numbered atomic ops
(2) int add fetch: 3
(8) int fetch sub: 2
(4) long xor fetch: 3
(5) long or fetch: 7
(10) unsigned fetch xor: 0
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 0
(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P4 Consumer got SCR1 post signal
P4 Consumer tries to lock MCR1
P4 Consumer successfully locked MCR1
P1 Producer unlocked MCR1
P5 Consumer tries to lock Sig21
P5 Consumer broadcasted Sig21
P5 Consumer unlocked Sig21
P5 Consumer waits for SCR1 post signal
P3 got Sig21
P3 unlocked Sig21
P4 Consumer successfully took 3 from CR1 list
Now there is 0 elements left in CR1 list
P4 Consumer unlocked MCR1

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 3 + -1 = 2$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P3 waits on Barrier bcr2
P6 got signal Sig21
P6 unlocked Sig21

P6 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 3 + -1 = 2$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P6 waits on Barrier bcr2

P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops

(2) int add fetch: 2

(8) int fetch sub: -1

(4) long xor fetch: 4

P3 passed through Barrier bcr2

P3 uses atomic variables

$\text{ivar1} + \text{ivar2} = 2 + -3 = -1$

$\text{uvar1} + \text{uvar2} = 7 + 7 = 14$

$\text{lvar1} + \text{lvar2} = 4 + 7 = 11$

(5) long or fetch: 7

(10) unsigned fetch xor: 7

(11) unsigned fetch or: 7

$\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

(13) long unsigned compare exchange success: 1

(14) long unsigned before exchange: 6 7

(14) long unsigned after exchange: 7 6

P4 tries to lock Sig21

P4 Consumer waits for Sig21

P4 Consumer got Sig21

P4 Consumer unlocked Sig21

P4 Consumer waits for SCR1 post signal

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P1 Producer unlocked MCR1

P2 Consumer got SCR1 post signal

P2 Consumer tries to lock MCR1

P2 Consumer successfully locked MCR1

P2 Consumer successfully took 4 from CR1 list

Now there is 0 elements left in CR1 list

P2 Consumer unlocked MCR1

P3 tries to lock Sig21

P3 waits for Sig21

P6 tries to lock Sig21

P6 waits for Sig21

P2 tries to lock Sig21

P2 Consumer waits for Sig21

P2 Consumer got Sig21

P2 unlocked Sig21

P2 uses atomic variables

$\text{ivar1} + \text{ivar2} = 2 + -3 = -1$

$\text{uvar1} + \text{uvar2} = 0 + 7 = 7$

$\text{lvar1} + \text{lvar2} = 4 + 7 = 11$

$\text{luvar1} + \text{luvar2} = 7 + 6 = 13$

P2 modifies atomic variables using numbered atomic ops

(2) int add fetch: -1

(8) int fetch sub: -3

(4) long xor fetch: 3

(5) long or fetch: 7

(10) unsigned fetch xor: 0

(11) unsigned fetch or: 7

(13) long unsigned compare exchange success: 0

(14) long unsigned before exchange: 7 7

(14) long unsigned after exchange: 7 7

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P1 Producer unlocked MCR1

P2 Consumer waits for SCR1 post signal

P4 Consumer got SCR1 post signal

P4 Consumer tries to lock MCR1

P4 Consumer successfully locked MCR1

P4 Consumer successfully took 5 from CR1 list

Now there is 0 elements left in CR1 list

P4 Consumer unlocked MCR1

P4 tries to lock Sig21

P4 Consumer waits for Sig21

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P5 Consumer got SCR1 post signal

P5 Consumer tries to lock MCR1

P5 Consumer successfully locked MCR1

P5 Consumer successfully took 6 from CR1 list

Now there is 0 elements left in CR1 list

P5 Consumer unlocked MCR1

P1 Producer unlocked MCR1

P1 Producer tries to lock MCR1

P1 Producer successfully locked MCR1

P1 Producer successfully added a new element number 0 to CR1 list

P1 Producer sent SCR1 post signal

P2 Consumer got SCR1 post signal

P2 Consumer tries to lock MCR1

P2 Consumer successfully locked MCR1

P2 Consumer successfully took 7 from CR1 list

P1 Producer unlocked MCR1

Now there is 0 elements left in CR1 list

P2 Consumer unlocked MCR1

P5 modifies atomic variables using numbered atomic ops

(2) int add fetch: -3

(8) int fetch sub: -2

(4) long xor fetch: 4

(5) long or fetch: 7

(10) unsigned fetch xor: 7
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 1
(14) long unsigned before exchange: 5 7
(14) long unsigned after exchange: 7 5
P5 Consumer tries to lock Sig21
P5 Consumer broadcasted Sig21
P5 Consumer unlocked Sig21
P5 Consumer waits for SCR1 post signal
P3 got Sig21
P3 unlocked Sig21

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -3 + 1 = -2$
P4 Consumer got Sig21
P4 Consumer unlocked Sig21
P4 Consumer waits for SCR1 post signal
P6 got signal Sig21
P6 unlocked Sig21

P6 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -3 + 1 = -2$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 5 = 12$

P3 waits on Barrier bcr2
 $\text{luvar1} + \text{luvar2} = 7 + 5 = 12$

P6 waits on Barrier bcr2
P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops
(2) int add fetch: -2
(8) int fetch sub: 1
(4) long xor fetch: 3
(5) long or fetch: 7
(10) unsigned fetch xor: 0
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 0
(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P3 passed through Barrier bcr2

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -2 + 3 = 1$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P2 tries to lock Sig21
P2 Consumer waits for Sig21
P2 Consumer got Sig21
P2 unlocked Sig21

P2 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -2 + 3 = 1$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P2 modifies atomic variables using numbered atomic ops
(2) int add fetch: 1
(8) int fetch sub: 3
(4) long xor fetch: 4
(5) long or fetch: 7
(10) unsigned fetch xor: 7
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 1
(14) long unsigned before exchange: 2 7
(14) long unsigned after exchange: 7 2
P2 Consumer waits for SCR1 post signal
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1
P5 Consumer got SCR1 post signal
P5 Consumer tries to lock MCR1
P5 Consumer successfully locked MCR1
P5 Consumer successfully took 8 from CR1 list
Now there is 0 elements left in CR1 list
P5 Consumer unlocked MCR1
P6 tries to lock Sig21
P6 waits for Sig21
P3 tries to lock Sig21
P3 waits for Sig21
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1

P5 modifies atomic variables using numbered atomic ops
(2) int add fetch: 3
(8) int fetch sub: 2
(4) long xor fetch: 3
(5) long or fetch: 7
(10) unsigned fetch xor: 0
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 0

(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P4 Consumer got SCR1 post signal
P4 Consumer tries to lock MCR1
P4 Consumer successfully locked MCR1
P4 Consumer successfully took 9 from CR1 list
Now there is 0 elements left in CR1 list
P5 Consumer tries to lock Sig21
P5 Consumer broadcasted Sig21
P5 Consumer unlocked Sig21
P4 Consumer unlocked MCR1
P5 Consumer waits for SCR1 post signal
P6 got signal Sig21
P6 unlocked Sig21

P6 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 3 + -1 = 2$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P6 waits on Barrier bcr2
P3 got Sig21
P3 unlocked Sig21

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 3 + -1 = 2$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P3 waits on Barrier bcr2
P3 passed through Barrier bcr2

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = 3 + -1 = 2$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops
(2) int add fetch: 2
(8) int fetch sub: -1
(4) long xor fetch: 4
(5) long or fetch: 7
(10) unsigned fetch xor: 7
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 1
(14) long unsigned before exchange: 6 7
(14) long unsigned after exchange: 7 6

P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1
P5 Consumer got SCR1 post signal
P5 Consumer tries to lock MCR1
P5 Consumer successfully locked MCR1
P5 Consumer successfully took 10 from CR1 list
Now there is 0 elements left in CR1 list
P5 Consumer unlocked MCR1
P4 tries to lock Sig21
P4 Consumer waits for Sig21
P4 Consumer got Sig21
P4 Consumer unlocked Sig21
P4 Consumer waits for SCR1 post signal
P3 tries to lock Sig21
P3 waits for Sig21
P6 tries to lock Sig21
P6 waits for Sig21
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1

P5 modifies atomic variables using numbered atomic ops
(2) int add fetch: -1
(8) int fetch sub: -3
(4) long xor fetch: 3
(5) long or fetch: 7
(10) unsigned fetch xor: 0
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 0
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1
(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P5 Consumer tries to lock Sig21
P2 Consumer got SCR1 post signal
P2 Consumer tries to lock MCR1
P2 Consumer successfully locked MCR1
P2 Consumer successfully took 11 from CR1 list
Now there is 0 elements left in CR1 list
P2 Consumer unlocked MCR1
P5 Consumer broadcasted Sig21
P5 Consumer unlocked Sig21
P5 Consumer waits for SCR1 post signal
P6 got signal Sig21
P6 unlocked Sig21

P6 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -1 + -2 = -3$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$

$\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P6 waits on Barrier bcr2
P3 got Sig21
P3 unlocked Sig21

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -1 + -2 = -3$
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 3 + 7 = 10$
 $\text{luvar1} + \text{luvar2} = 7 + 7 = 14$

P3 waits on Barrier bcr2
P3 passed through Barrier bcr2

P3 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -1 + -2 = -3$
P6 passed through Barrier bcr2

P6 modifies atomic variables using numbered atomic ops

(2) int add fetch: -3
(8) int fetch sub: -2
(4) long xor fetch: 4
(5) long or fetch: 7
(10) unsigned fetch xor: 7
(11) unsigned fetch or: 7
 $\text{uvar1} + \text{uvar2} = 7 + 7 = 14$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 6 + 7 = 13$

(13) long unsigned compare exchange success: 1
(14) long unsigned before exchange: 6 7
(14) long unsigned after exchange: 7 6
P2 tries to lock Sig21
P2 Consumer waits for Sig21
P2 Consumer got Sig21
P2 unlocked Sig21

P2 uses atomic variables
 $\text{ivar1} + \text{ivar2} = -3 + 1 = -2$
 $\text{uvar1} + \text{uvar2} = 0 + 7 = 7$
 $\text{lvar1} + \text{lvar2} = 4 + 7 = 11$
 $\text{luvar1} + \text{luvar2} = 7 + 6 = 13$

P2 modifies atomic variables using numbered atomic ops

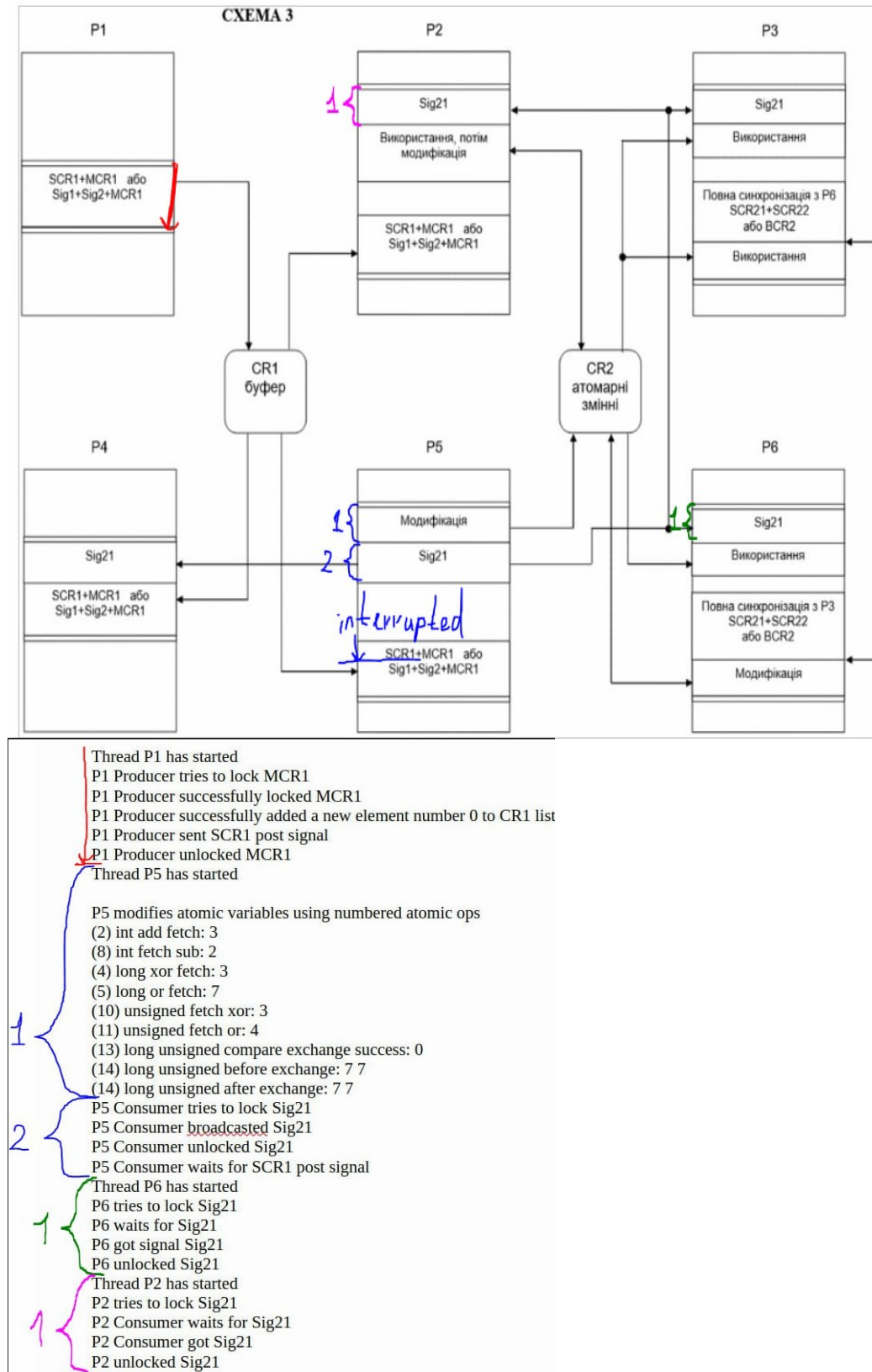
(2) int add fetch: -2
(8) int fetch sub: 1
(4) long xor fetch: 3
(5) long or fetch: 7
(10) unsigned fetch xor: 0

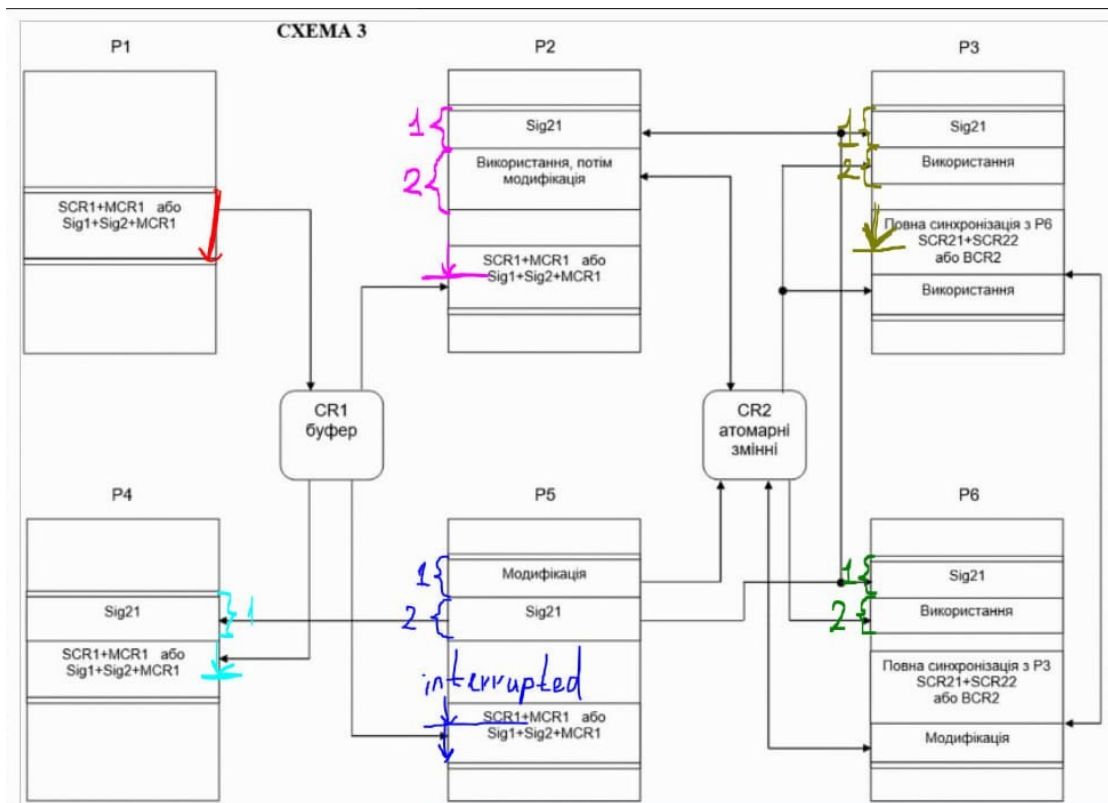
(11) unsigned fetch or: 7
(13) long unsigned compare exchange success: 0
(14) long unsigned before exchange: 7 7
(14) long unsigned after exchange: 7 7
P2 Consumer waits for SCR1 post signal
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1
P4 Consumer got SCR1 post signal
P4 Consumer tries to lock MCR1
P4 Consumer successfully locked MCR1
P4 Consumer successfully took 12 from CR1 list
Now there is 0 elements left in CR1 list
P4 Consumer unlocked MCR1
P3 tries to lock Sig21
P3 waits for Sig21
P6 tries to lock Sig21
P6 waits for Sig21
P4 tries to lock Sig21
P4 Consumer waits for Sig21
P4 Consumer got Sig21
P4 Consumer unlocked Sig21
P4 Consumer waits for SCR1 post signal
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P2 Consumer got SCR1 post signal
P2 Consumer tries to lock MCR1
P2 Consumer successfully locked MCR1
P1 Producer unlocked MCR1
P2 Consumer successfully took 13 from CR1 list
Now there is 0 elements left in CR1 list
P2 Consumer unlocked MCR1
P1 Producer tries to lock MCR1
P1 Producer successfully locked MCR1
P1 Producer successfully added a new element number 0 to CR1 list
P1 Producer sent SCR1 post signal
P1 Producer unlocked MCR1
P4 Consumer got SCR1 post signal
P4 Consumer tries to lock MCR1
P4 Consumer successfully locked MCR1
P2 tries to lock Sig21
P2 Consumer waits for Sig21
P4 Consumer successfully took 14 from CR1 list
Now there is 0 elements left in CR1 list
P4 Consumer unlocked MCR1

^C - Завершення роботи Ctrl + C

Базовий аналіз протоколу:

Перевіримо чи вірно паралельні потоки виконують поставлену підзадачу синхронізації. Співставимо printf повідомлення з наданою схемою.





P2 uses atomic variables
 $ivar1 + ivar2 = 3 + -1 = 2$
 $uvar1 + uvar2 = 7 + 7 = 14$
 $lvar1 + lvar2 = 3 + 7 = 10$
 $luvar1 + luvar2 = 7 + 7 = 14$

P2 modifies atomic variables using numbered atomic ops
 (2) int add fetch: 2
 (8) int fetch sub: -1
 (4) long xor fetch: 4
 (5) long or fetch: 7
 (10) unsigned fetch xor: 7
 (11) unsigned fetch or: 7
 (13) long unsigned compare exchange success: 1
 (14) long unsigned before exchange: 2 7
 (14) long unsigned after exchange: 7 2

P2 Consumer waits for SCR1 post signal

Thread P4 has started

P4 tries to lock Sig21

P4 Consumer waits for Sig21

P4 Consumer got Sig21

P4 Consumer unlocked Sig21

P4 Consumer waits for SCR1 post signal

Thread P3 has started

P3 tries to lock Sig21

P3 waits for Sig21

P3 got Sig21

P3 unlocked Sig21

P3 uses atomic variables

$ivar1 + ivar2 = 2 + -3 = -1$

$uvar1 + uvar2 = 0 + 7 = 7$

$lvar1 + lvar2 = 4 + 7 = 11$

$luvar1 + luvar2 = 7 + 2 = 9$

P3 waits on Barrier bcr2

P5 Consumer got SCR1 post signal

P5 Consumer tries to lock MCR1

P5 Consumer successfully locked MCR1

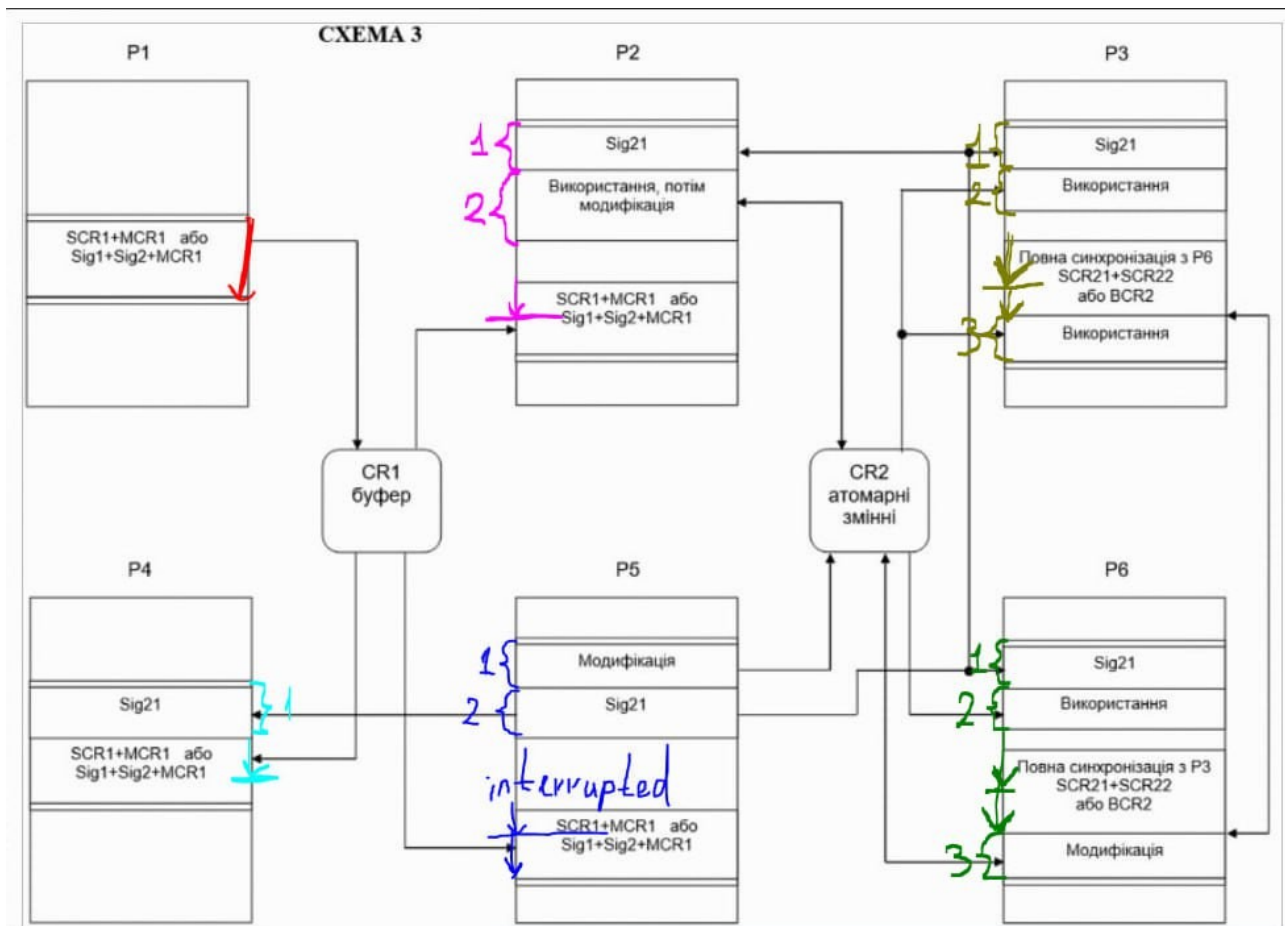
P6 uses atomic variables



$ivar1 + ivar2 = 2 + -3 = -1$


$uvar1 + uvar2 = 0 + 7 = 7$


$lvar1 + lvar2 = 4 + 7 = 11$


$luvar1 + luvar2 = 7 + 2 = 9$




 P6 waits on Barrier bcr2
 P6 passed through Barrier bcr2

 P6 modifies atomic variables using numbered atomic ops
 (2) int add fetch: -1
 (8) int fetch sub: -3

 P3 passed through Barrier bcr2

 P3 uses atomic variables
 (4) long xor fetch: 3

 P5 Consumer successfully took 0 from CR1 list

Висновок

Як бачимо на наведених зверху діаграмах, послідовність дій потоків, співпадає з поставленою перед ними задачею.