OS project checkpoint 5

105021120 朱世耘

function address:

```
      Value  Global                                Global Defined In Module
      -----  -----------------------------------   ------------------------
C:    00000014  _Car1                               testparking
C:    00000051  _Car2                               testparking
C:    0000008E  _Car3                               testparking
C:    000000CB  _Car4                               testparking
C:    00000108  _Car5                               testparking
C:    00000145  _output                             testparking
C:    00000BEB  _main                               testparking
C:    00000C18  __sdcc_gsinit_startup               testparking
C:    00000C1C  __mcs51_genRAMCLEAR                 testparking
C:    00000C1D  __mcs51_genXINIT                    testparking
C:    00000C1E  __mcs51_genXRAMCLEAR                testparking
C:    00000C1F  _timer0_ISR                         testparking
C:    00000C23  _now                                preemptive
C:    00000C27  _delay                              preemptive
C:    00000C47  _thread_manager                     preemptive
C:    00000C7A  _myTimer0Handler                    preemptive
C:    00000CD2  _Bootstrap                          preemptive
C:    00000D1E  _ThreadCreate                       preemptive
C:    00000D84  _ThreadYield                        preemptive
C:    00000DCF  _ThreadExit                         preemptive
```

data address:

```
 Value  Global                                Global Defined In Module
 -----  -----------------------------------   ------------------------
00000000  .__.ABS.                             preemptive
00000020  _mutex                               testparking
00000021  _thread_count                        preemptive
00000022  _parklot                             testparking
00000024  _bitmap                              preemptive
00000029  _manager_ID                          preemptive
0000002A  _tmp1                                preemptive
0000002B  _tmp2                                preemptive
0000002C  _i                                   preemptive
0000002D  _count                               preemptive
0000002E  _x                                   testparking
00000030  _time                                preemptive
00000031  _log                                 testparking
0000004A  _ID                                  preemptive
0000005C  _ssp                                 preemptive
0000006C  _d_time                              preemptive
0000007D  _printer                             testparking
0000007E  _cnt                                 preemptive
0000007F  _var                                 preemptive
```

log[15] : from 31 to 3F, record every cars' arrived/leave/location

time : 30, increase when thread(main) get the power

parklot[2] : 22,23, record which car is in the parking lot now

bitmap[4] : from 24 to 27, record every thread status 0:empty 1:using 7:delaying

key function:

now( ) and delay( )

```
unsigned char now(void){
    return time;
    // time++ when thread[1](main) get the power
}
void delay(unsigned char n){
    d_time[ID]=now()+n;
    // set bitmap[]=7 represent delay
    bitmap[ID]=7;
    ThreadYield();
}
```

TimeroHandler( )

```
void myTimer0Handler(void){
    SAVESTATE;
    // check if any delay thread times up
    for (i=0; i<MAXTHREADS; i++) {
        if ((bitmap[i]==7) && (d_time[i]==time)) {
            bitmap[i]=1;
        }
    }
    tmp1 = ID;
    ID = manager_ID;
    RESTORESTATE;
    ID = tmp1;

}
```

ThreadYield( )

```
void ThreadYield(void) {
    EA = 0;
    SAVESTATE;
    // find next avalible thread
    do {
        ID++;
        ID = ID % MAXTHREADS;
        if(bitmap[ID]==1){
            break;
        }
    } while (1);
    // check if time should increase
    if (ID==1) {
        cnt++;
        if (cnt==3) {
            cnt=0;
            time++;
        }
    }
    RESTORESTATE;
    EA = 1;
}
```

time++ when thread(main) get power every three times

Car1( ) ,same as other cars

```
// Car 1 to 5, find empty parklot, record in log[], leave
void Car1(void) {
    log[0] = now();
    SemaphoreWait(mutex);
    if (parklot[0]==0) {
        parklot[0]=1;
        log[2]=0;
    }
    else{
        parklot[1]=1;
        log[2]=1;
    }
    SemaphoreSignal(mutex);
    delay(2);
    log[1] = now();
    parklot[(log[2])]=0;
    SemaphoreSignal(printer);
    ThreadExit();
}
```

output( )

```
// output log[]
void output(){
    // wait until five cars have leaved
    SemaphoreWait(printer);
    SemaphoreWait(printer);
    SemaphoreWait(printer);
    SemaphoreWait(printer);
    SemaphoreWait(printer);

    TMOD |= 0x20;
    TH1 = -6;
    SCON = 0x50;
    TR1 = 1;

    // output car i:arrive/leave/location
    print('\0');
    print('a');
```

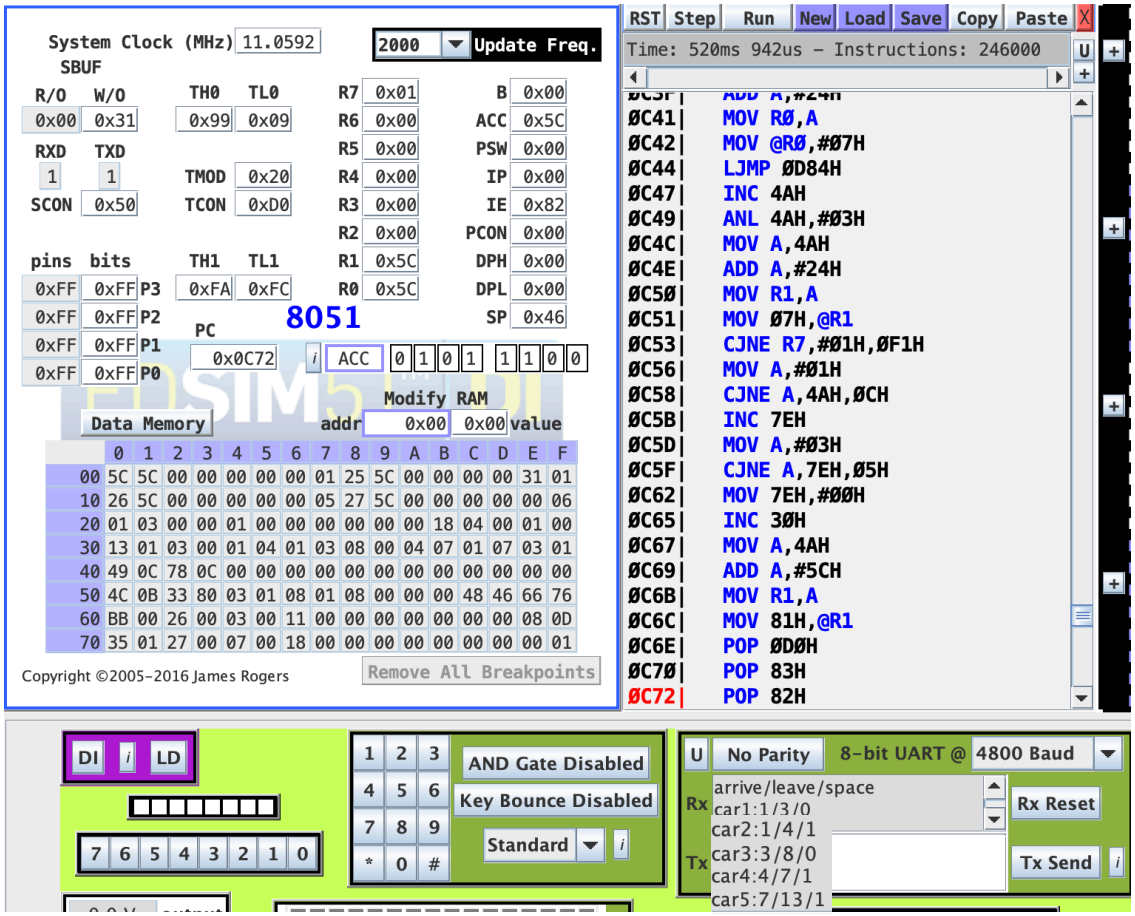
main( )

```
void main(void) {
    // initialize Semaphore and parklot
    SemaphoreCreate(mutex, 1);
    SemaphoreCreate(printer, 0);
    parklot[0]=0;
    parklot[1]=0;

    ThreadCreate(Car1);
    ThreadCreate(Car2);
    ThreadCreate(Car3);
    ThreadCreate(Car4);
    ThreadCreate(Car5);
    output();

}
```

execution:
Let car1 delay(2), car2 delay(3), car3 delay(5), car4 delay(3), car5 delay(6)
therefore, two parking lots should be:
parklot[0]: 1~3 car1, 3~8 car3
parklot[1]: 1~4 car2, 4~7 car4, 7~13 car5



0x31 to 0x3F store log[15]

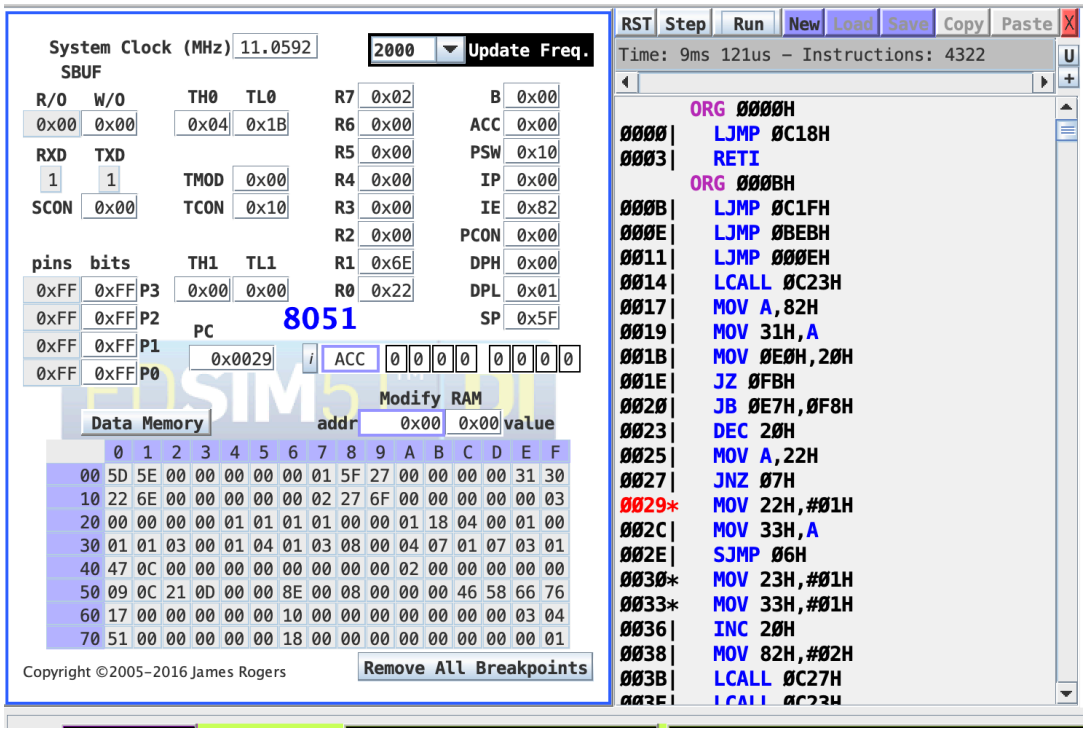
	arrive	leave	space
car1:	1	3	0
car2:	1	4	1
car3:	3	8	0
car4:	4	7	1
car5:	7	13	1

Observe semaphore:

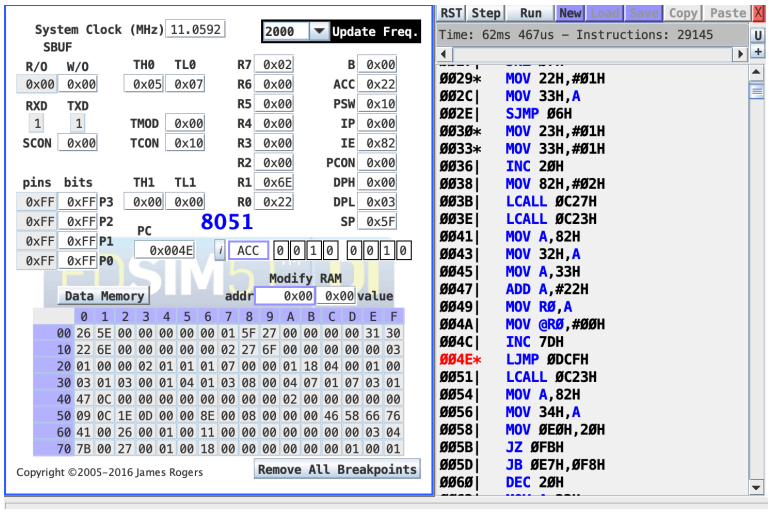
there are three semaphore,
mutex: at 20, initial to be 1, change when enter or leave critical section
thread_count: at 21, initial to be 4, change when threadcreate and threadexit
printer: at 7D, initial to be 0, increase when car leave, decrease when outputing

```
// Car 1 to 5, find empty parklot, record in log[], leave
void Car1(void) {
    log[0] = now();
    SemaphoreWait(mutex);
    if (parklot[0]==0) {
        parklot[0]=1;
        log[2]=0;
    }
    else{
        parklot[1]=1;
        log[2]=1;
    }
    SemaphoreSignal(mutex);
    delay(2);
    log[1] = now();
    parklot[(log[2])]=0;
    SemaphoreSignal(printer);
    ThreadExit();
}
```

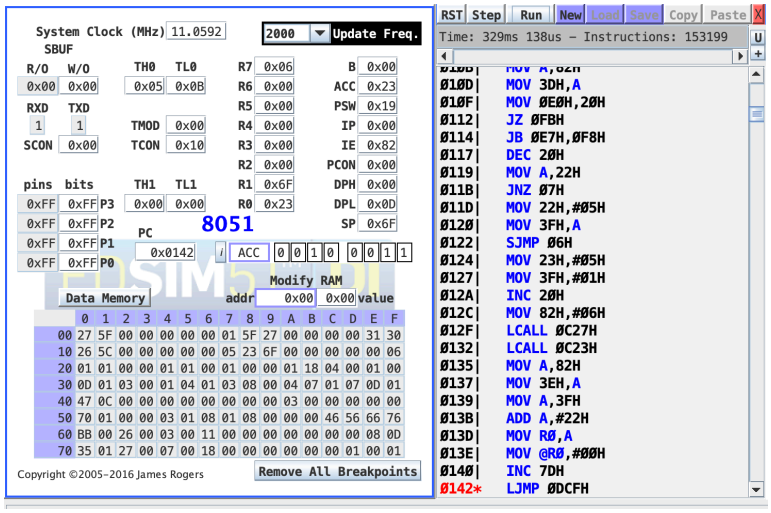
Car1 is doing critical section, mutex=0



Car1 finished, printer++



all the car have leaved, printer is large enough to output



outputing, printer=0