```
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\109062119-ppc4> make clean
del *.hex *.ihx *.lnk *.lst *.map *.mem *.rel *.rst *.sym *.asm *.lk
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\109062119-ppc4> make
sdcc -c  test3threads.c
test3threads.c:77: warning 158: overflow in implicit constant conversion
sdcc -c  preemptive.c
preemptive.c:157: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
preemptive.c:250: warning 158: overflow in implicit constant conversion
sdcc  -o test3threads.hex test3threads.rel preemptive.rel
PS C:\Users\josh9\OneDrive\桌面\大三上\OS\109062119-ppc4>
```

Producer1:

0xFF 0xFF P2 0xFF 0xFF P1 0xFF 0xFF P0 — 8051 — PC 0x0017 — i — PSW 0 0 0 0 1 0 0 0 — SP 0x4F

Modify RAM

Data Memory addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	0C	01	01	00	00	10	31	01	00	00	03	46	46	20
10	30	31	01	00	00	36	36	01	00	00	00	00	00	00	00	00
20	3F	00	00	00	1A	01	00	35	41	00	00	01	01	36	00	00
30	46	56	66	00	01	02	07	02	00	00	41	02	02	04	00	00
40	B3	00	00	00	02	00	90	30	30	0C	01	01	00	00	10	00
50	14	00	00	00	00	00	09	30	30	0C	01	01	00	00	10	00
60	5E	00	00	00	00	00	11	30	30	0C	01	01	00	00	10	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

opyright ©2005-2022 James Rogers — Remove All Breakpoints

```
ØØ11I  LJMP ØØØEH
ØØ14I  MOV 28H,#41H
ØØ17*  MOV A,26H
ØØ19I  JB ØE7H,ØFBH
ØØ1CI  JZ ØF9H
ØØ1EI  DEC 26H
ØØ20I  MOV A,2CH
ØØ22I  JB ØE7H,ØFBH
ØØ25I  JZ ØF9H
ØØ27I  DEC 2CH
ØØ29I  MOV A,2BH
ØØ2BI  JB ØE7H,ØFBH
ØØ2EI  JZ ØF9H
```

Data Memory addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	0C	01	01	00	00	10	31	32	00	00	03	42	42	00
10	30	31	01	00	00	31	31	01	00	00	00	00	00	00	00	00
20	3F	00	00	00	1A	01	00	41	42	00	01	01	00	31	00	00
30	46	56	66	00	01	02	07	02	00	00	41	02	02	04	00	00
40	B3	00	00	00	02	00	90	30	30	0C	01	01	00	00	10	00
50	17	00	00	00	00	00	08	30	30	0C	01	01	00	00	10	00
60	66	00	00	00	00	00	90	30	30	0C	01	01	00	00	10	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2022 James Rogers — Remove All Breakpoints

```
ØØ4CI  RLC A
ØØ4DI  SUBB A,ØEØH
ØØ4FI  MOV R7,A
ØØ5ØI  MOV 28H,R6
ØØ52I  MOV C,ØØH
ØØ54I  MOV ØAFH,C
ØØ56I  INC 2BH
ØØ58I  INC 2AH
ØØ5AI  INC 25H
ØØ5C*  SJMP ØB9H
```

Produce2:

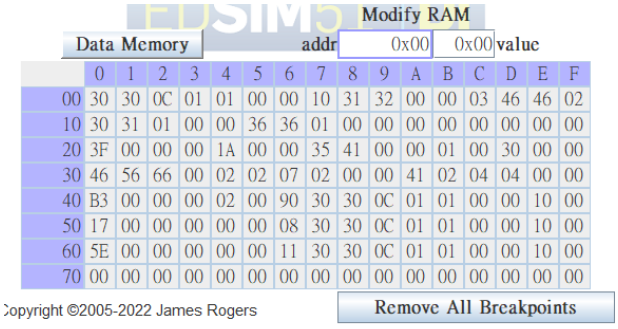
Modify RAM

Data Memory addr 0x00 0x00 value

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	0C	01	01	00	00	10	31	32	00	00	03	4D	4D	02
10	30	31	01	00	00	33	33	01	00	00	00	00	00	00	00	00
20	3F	00	00	00	1A	01	00	32	41	00	00	01	01	30	00	00
30	46	56	66	00	02	02	07	02	00	00	41	02	04	04	00	00
40	B3	00	00	00	02	00	90	30	30	0C	01	01	00	00	10	00
50	17	00	00	00	00	00	08	30	30	0C	01	01	00	00	10	00
60	5E	00	00	00	00	00	11	30	30	0C	01	01	00	00	10	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

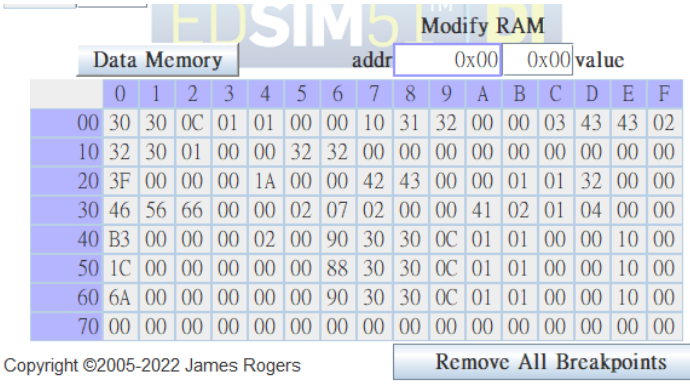
opyright ©2005-2022 James Rogers — Remove All Breakpoints

```
ØØ5AI  INC 25H
ØØ5CI  SJMP ØB9H
ØØ5EI  MOV 2DH,#3ØH
ØØ61*  MOV A,25H
ØØ63*  JB ØE7H,ØFBH
ØØ66I  JZ ØF9H
ØØ68I  DEC 25H
ØØ6AI  MOV A,2CH
ØØ6CI  JB ØE7H,ØFBH
```

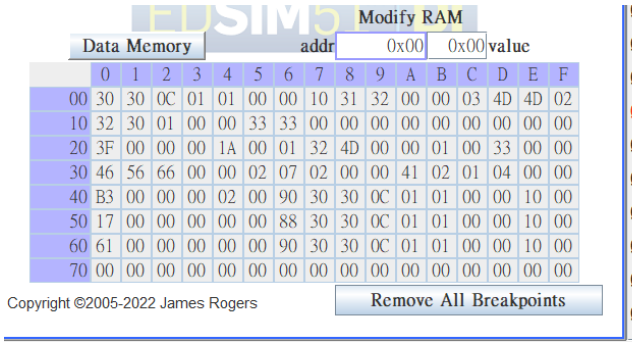



```
ØØ66I  JZ ØF9H
ØØ68I  DEC 25H
ØØ6AI  MOV A,2CH
ØØ6CI  JB ØE7H,ØFBH
ØØ6FI  JZ ØF9H
ØØ71I  DEC 2CH
ØØ73I  MOV A,2BH
ØØ75I  JB ØE7H,ØFBH
ØØ78I  JZ ØF9H
ØØ7AI  DEC 2BH
```

Consumer:




```
ØØAB*  MOV 8DH,#ØFAH
ØØAEI  MOV 98H,#5ØH
ØØB1I  SETB 8EH
ØØB3I  MOV A,2AH
ØØB5I  JB ØE7H,ØFBH
ØØB8I  JZ ØF9H
ØØBAI  DEC 2AH
ØØBCI  MOV A,2BH
ØØBEI  JB ØE7H,ØFBH
ØØC1I  JZ ØF9H
```




```
ØØB5I  JB ØE7H,ØFBH
ØØB8I  JZ ØF9H
ØØBAI  DEC 2AH
ØØBC*  MOV A,2BH
ØØBEI  JB ØE7H,ØFBH
ØØC1I  JZ ØF9H
ØØC3I  DEC 2BH
ØØC5I  SETB Ø2H
ØØC7I  JBC ØAFH,Ø2H
ØØCAI  CLR Ø2H
```

3.

```
ThreadCreate( Producer1 );
ThreadCreate( Producer2 );
```

```
0123456789012345
```

```
ThreadCreate( Producer2 );
ThreadCreate( Producer1 );
Consumer();
```

```
ABCDEFGHIJKLMNOPQRSTUVWXY
```

Only one kind of output will be shown.

test3threads.c > conca(x)

```
void Producer1(void) {
        Token = 'A';
        while (1) {
                SemaphoreWaitBody(abc, conca(__COUNTER__) );
                SemaphoreWaitBody(empty, conca(__COUNTER__) );
                SemaphoreWaitBody(mutex, conca(__COUNTER__) );

                __critical{
                        buffer = Token;
                        Token = ( Token == 'Z' ) ? 'A' :  Token + 1;

                }

                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                SemaphoreSignal(num);

        }
}
void Producer2(void) {
        Token1 = '0';
        while (1) {
                SemaphoreWaitBody(num, conca(__COUNTER__) );
                SemaphoreWaitBody(empty, conca(__COUNTER__) );
                SemaphoreWaitBody(mutex, conca(__COUNTER__) );
                __critical{

                        buffer = Token1;
                        Token1 = ( Token1 == '9' ) ? '0' :  Token1 + 1;

                }

                SemaphoreSignal(mutex);
                SemaphoreSignal(full);
                SemaphoreSignal(abc);

        }
}
```

```
0A1B2C3D4E
```

number and letter output by turns.