

## Programming Project Checkpoint 2

IPNS 106010018 Yen-Fong Li

### Testpreempt.map

Value	Global	Global Defined In Module
C: 00000014	_Producer	testpreempt
C: 00000044	_Consumer	testpreempt
C: 00000072	_main	testpreempt
C: 00000084	__sdcc_gsinit_startup	testpreempt
C: 00000088	_mcs51_genRAMCLEAR	testpreempt
C: 00000089	_mcs51_genXINIT	testpreempt
C: 0000008A	_mcs51_genXRAMCLEAR	testpreempt
C: 0000008B	_timer0_ISR	testpreempt
C: 0000008F	_Bootstrap	preemptive
C: 000000B5	_ThreadCreate	preemptive
C: 00000134	_ThreadYield	preemptive
C: 000001CA	_myTimer0Handler	preemptive
C: 0000025A	_ThreadExit	preemptive

From testpreempt.map, we can find out that the address of ThreadCreate() function is at 000000B5, so set checkpoint at 0085 and then run Edsim51.

System Clock (MHz) 11.0592 10000 Update Freq.

SBUF

R/O W/O TH0 TL0 R7 0x00 B 0x00

0x00 0x00 0x02 0x1D R6 0x00 ACC 0x00

RXD TXD TMOD 0x00 R5 0x00 PSW 0x00

1 1 TCON 0x10 R4 0x00 IP 0x00

SCON 0x00 R3 0x00 IE 0x82

pins bits TH1 TL1 R2 0x00 PCON 0x00

0xFF 0xFF P3 0x00 0x00 R1 0x30 DPH 0x00

0xFF 0xFF P2 PC 8051 R0 0x30 DPL 0x14

0xFF 0xFF P1 PSW 0 0 0 0 0 0 0 0 SP 0x41

0xFF 0xFF P0 0x00B5

Modify RAM

addr 0x00 0x00 value

Data Memory

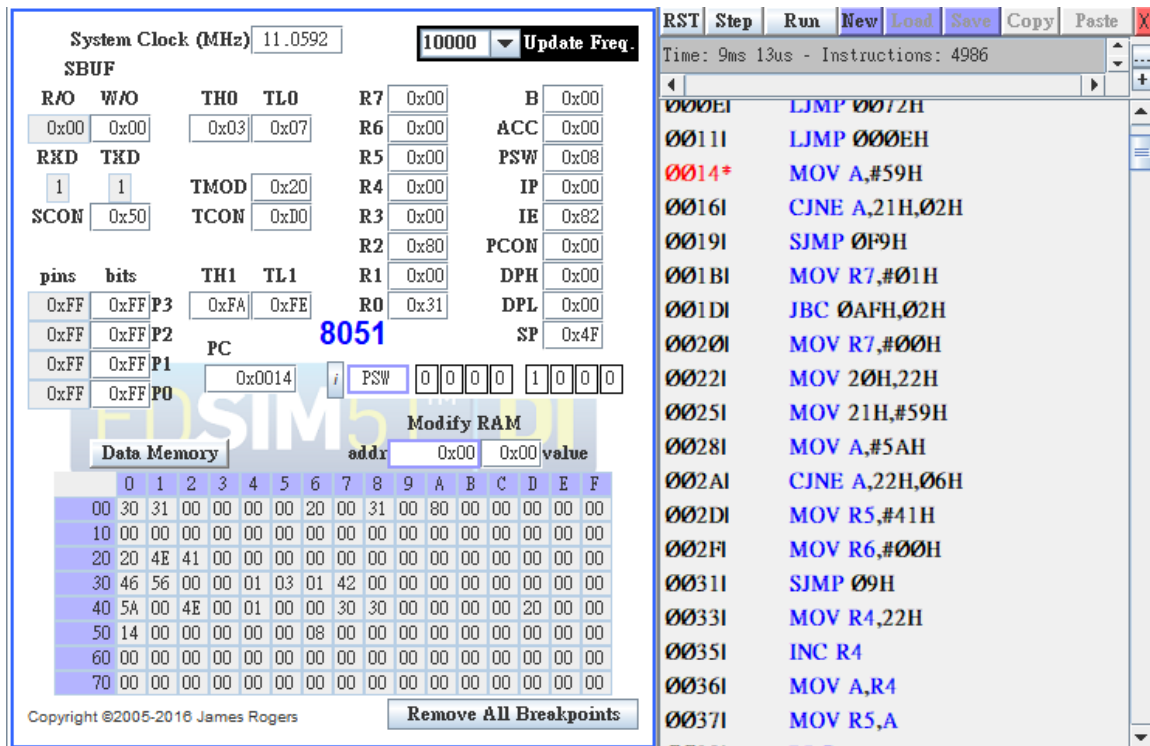
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	30	30	00	00	00	00	00	00	00	A0	00	80	00	00	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	20	4E	41	00	00	00	00	00	00	00	00	00	00	00	00	00
30	46	00	00	00	00	01	00	0A	00	00	00	00	00	00	00	00
40	81	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Copyright ©2005-2016 James Rogers Remove All Breakpoints

```
#06A| RRC A
#06B| MOV #AFH, C
#06D| JBC 99H, #E5H
#070| SJMP #FBH
#072| MOV 20H, #20H
#075| MOV 21H, #4EH
#078| MOV 22H, #41H
#07A| MOV DPH, #14H
#07E| LCALL #B5H
#081| LJMP #44H
#084| LJMP #8Fh
#087| RET
#088| RET
#089| RET
#08A| RET
#08B| LJMP #1CAH
#08E| RETI
#08F| MOV 35H, #00H
#092| MOV 89H, #00H
#095| MOV #A8H, #82H
#098| SETB 8CH
```

0x40 save the return address, after ThreadCreate(), the program can go back and continue

## Producer



From testcoop.map, we can tell that the address of Producer is at 0009. Set checkpoint to it and run Edsim51.

To know whether the Producer is running, it can be observed from the address 0x34, which points to the current\_id. (current thread id) The value of it is 01, which means Producer is running.

## Consumer

The screenshot shows the Edsim51 microsimulator interface. The top panel displays system settings: System Clock (MHz) at 11.0592, a frequency dropdown at 10000, and an 'Update Freq.' button. Below this, the SBUF register is shown with R/O and W/O bits set to 0x00. The TH0 and TL0 registers are at 0x04 and 0x1D. The RxD and TxD pins are set to 1. The TMOD register is at 0x00 and TCON is at 0x10. The SCON register is at 0x00. The pins and bits section shows P3, P2, P1, and P0 all set to 0xFF. The TH1 and TL1 registers are at 0x00. The PC register is at 0x0044. The PSW register is at 8051. The R0 register is at 0x30. The R1 register is at 0x30. The R2 register is at 0x00. The R3 register is at 0x00. The R4 register is at 0x00. The R5 register is at 0x00. The R6 register is at 0x00. The R7 register is at 0x01. The B register is at 0x00. The ACC register is at 0x31. The PSW register is at 0x81. The IP register is at 0x00. The IE register is at 0x82. The PCON register is at 0x00. The DPH register is at 0x00. The DPL register is at 0x01. The SP register is at 0x3F. The Data Memory section shows a table of memory addresses from 0 to 70. The Modify RAM section shows a table of memory addresses from 0 to 70. The instruction list on the right shows the following instructions:

Address	Instruction
003E	MOV A,R7
003F	RRC A
0040	MOV 0AFH,C
0042	SJMP 0D0H
0044*	MOV R6,89H
0046	MOV R7,#00H
0048	ORL 06H,#20H
004B	MOV 89H,R6
004D	MOV 8DH,#0FAH
0050	MOV 98H,#50H
0053	SETB 8EH
0055	MOV A,#4EH
0057	CJNE A,21H,02H
005A	SJMP 0F9H
005C	MOV R7,#01H
005E	JBC 0AFH,02H
0061	MOV R7,#00H
0063	MOV 99H,20H
0066	MOV 21H,#4EH

From testcoop.map, we can tell that the address of Producer is at 0009. Set checkpoint to it and run Edsim51.

To know whether the Consumer is running, it can be observed from the address 0x34, which points to the current\_id. (current thread id) The value of it is 00, which means Producer is running.

## myTimer0Handler (Interrupt triggered)

When an interrupt is triggered, the function myTimer0Handler() will be called. From testpreempt.map, we know that the address of myTimer0Handler() is 01cA. Set a breakpoint to it and then run edsim51.

Since I wrote parts of my code of myTimer0Handler() in C between SAVESTATE and RESTORESTATE, it is likely to use R0 and R1, any code that modifies R0-R7 will trash them. I had to save them to the stack so I could restore their value.

In that case, when interrupt is triggering, we can notice that the top of the stack will have the value of R0-R7.

The screenshot displays the EDSIM51 microcontroller simulator interface. The top section shows the System Clock (MHz) at 11.0592 and a frequency of 10000. The left panel contains various registers and control bits, including SBUF, R7, R6, R5, R4, R3, R2, R1, R0, TH0, TL0, TMOD, TCON, TH1, TL1, PC, PSW, and P0-P3. The right panel shows the assembly code being executed, starting with 01D6: MOV A,34H and ending with 01F2: MOV A,R7. The bottom panel shows the Data Memory (RAM) with addresses 0 to 70. A red circle highlights the R0 register, which contains the value 0x31. Another red circle highlights the memory address 0x31 in the Data Memory, which contains the value 0x31. The PSW register is also highlighted with a red circle and contains the value 0x00.

R7	R6	R5	R4	R3	R2	R1	R0	TH0	TL0	TMOD	TCON	TH1	TL1	PC	PSW
0x01	0x00	0x46	0x46	0x00	0x80	0x30	0x31	0x01	0x09	0x20	0x00	0xFA	0xFE	0x01F2	0x00

Address	Value
0x00	0x00
0x01	0x00
0x02	0x00
0x03	0x00
0x04	0x00
0x05	0x00
0x06	0x00
0x07	0x00
0x08	0x00
0x09	0x00
0x0A	0x00
0x0B	0x00
0x0C	0x00
0x0D	0x00
0x0E	0x00
0x0F	0x00
0x10	0x00
0x11	0x00
0x12	0x00
0x13	0x00
0x14	0x00
0x15	0x00
0x16	0x00
0x17	0x00
0x18	0x00
0x19	0x00
0x1A	0x00
0x1B	0x00
0x1C	0x00
0x1D	0x00
0x1E	0x00
0x1F	0x00
0x20	0x00
0x21	0x00
0x22	0x00
0x23	0x00
0x24	0x00
0x25	0x00
0x26	0x00
0x27	0x00
0x28	0x00
0x29	0x00
0x2A	0x00
0x2B	0x00
0x2C	0x00
0x2D	0x00
0x2E	0x00
0x2F	0x00
0x30	0x00
0x31	0x31
0x32	0x00
0x33	0x00
0x34	0x00
0x35	0x00
0x36	0x00
0x37	0x00
0x38	0x00
0x39	0x00
0x3A	0x00
0x3B	0x00
0x3C	0x00
0x3D	0x00
0x3E	0x00
0x3F	0x00
0x40	0x00
0x41	0x00
0x42	0x00
0x43	0x00
0x44	0x00
0x45	0x00
0x46	0x00
0x47	0x00
0x48	0x00
0x49	0x00
0x4A	0x00
0x4B	0x00
0x4C	0x00
0x4D	0x00
0x4E	0x00
0x4F	0x00
0x50	0x00
0x51	0x00
0x52	0x00
0x53	0x00
0x54	0x00
0x55	0x00
0x56	0x00
0x57	0x00
0x58	0x00
0x59	0x00
0x5A	0x00
0x5B	0x00
0x5C	0x00
0x5D	0x00
0x5E	0x00
0x5F	0x00
0x60	0x00
0x61	0x00
0x62	0x00
0x63	0x00
0x64	0x00
0x65	0x00
0x66	0x00
0x67	0x00
0x68	0x00
0x69	0x00
0x6A	0x00
0x6B	0x00
0x6C	0x00
0x6D	0x00
0x6E	0x00
0x6F	0x00
0x70	0x00

```
01D6: MOV A,34H
01D8: ADD A,#30H
01DA: MOV R0,A
01DB: MOV @R0,81H
01DD: MOV A,R0
01DE: PUSH 0E0H
01E0: MOV A,R1
01E1: PUSH 0E0H
01E3: MOV A,R2
01E4: PUSH 0E0H
01E6: MOV A,R3
01E7: PUSH 0E0H
01E9: MOV A,R4
01EA: PUSH 0E0H
01EC: MOV A,R5
01ED: PUSH 0E0H
01EF: MOV A,R6
01F0: PUSH 0E0H
01F2: MOV A,R7
```

## Explanations

This checkpoint is similar with the last checkpoint. There are some parts to change.

1. Don't need ThreadYield()
2. Plus \_\_critical  
Since we implement Producer-Consumer Problem by preemptive way, we need to add \_\_critical to the part that are shared, preventing from multiple accesses.
3. Plus EA  
EA is used to disable interrupt and enable interrupt by changing its value to 0 or 1.