

Programming Project Checkpoint 1

111062311 林哲宇

1. Screenshot for compilation:

```
jeffreylin0909@DESKTOP-Q29MBHF:/mnt/c/Users/林哲宇/OneDrive/桌面/OS/OS check point 1/ppc1$ make
sdcc -c testcoop.c
testcoop.c:63: warning 158: overflow in implicit constant conversion
sdcc -c cooperative.c
cooperative.c:196: warning 85: in function ThreadCreate unreferenced function argument : 'fp'
sdcc -o testcoop.hex testcoop.rel cooperative.rel
jeffreylin0909@DESKTOP-Q29MBHF:/mnt/c/Users/林哲宇/OneDrive/桌面/OS/OS check point 1/ppc1$ ls
Makefile      cooperative.h  cooperative.rst  testcoop.c  testcoop.lst  testcoop.rel
cooperative.asm cooperative.lst  cooperative.sym  testcoop.hex  testcoop.map  testcoop.rst
cooperative.c  cooperative.rel  testcoop.asm   testcoop.lk  testcoop.mem  testcoop.sym
jeffreylin0909@DESKTOP-Q29MBHF:/mnt/c/Users/林哲宇/OneDrive/桌面/OS/OS check point 1/ppc1$ |
```

- ▶ The warning message is because of setting TH1 to negative number (-6 for UART baud rate), and it's totally safe.
- ▶ The warning message is because of using DPH/DPL instead of identifier of parameter, and it's totally safe.
- ▶ Files generated after compilation (including .hex and .map).

Note: for better understanding for following explanation, here's the variable address map of my code:

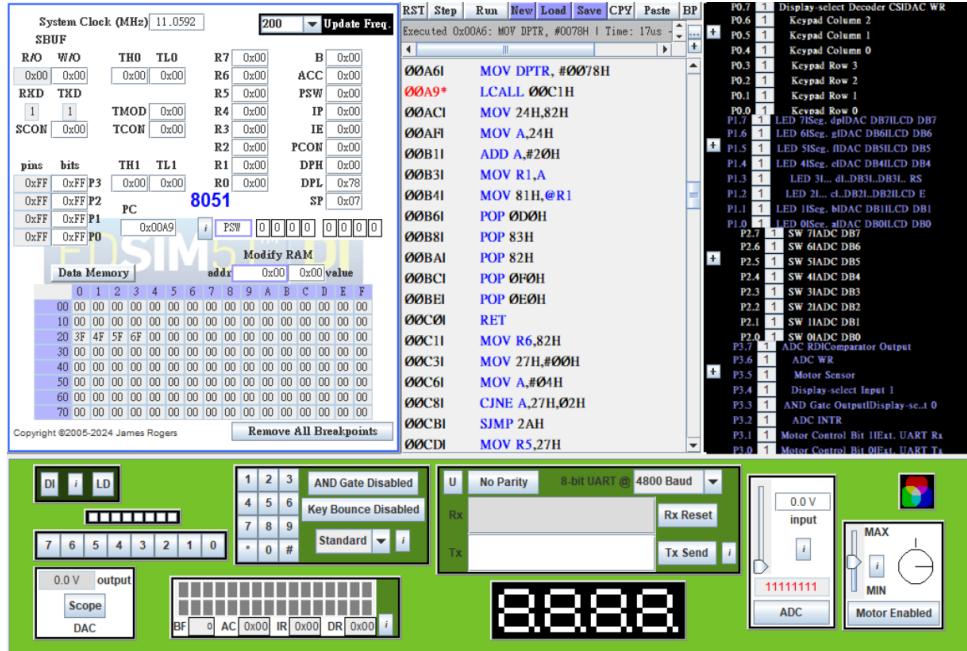
00000020	_T_SP (20~23,1 address space each entry)	cooperative
00000024	_current_T	cooperative
00000025	_tmp0	cooperative
00000026	_tmp1	cooperative
00000027	_tmp2	cooperative
00000028	_bitmap	cooperative
0000002C	_b_start	testcoop
0000002D	_b_end	testcoop
0000002E	_in_counter	testcoop
00000030	_buffer (30~3F,1 address space each entry)	testcoop

And here's the function address map:

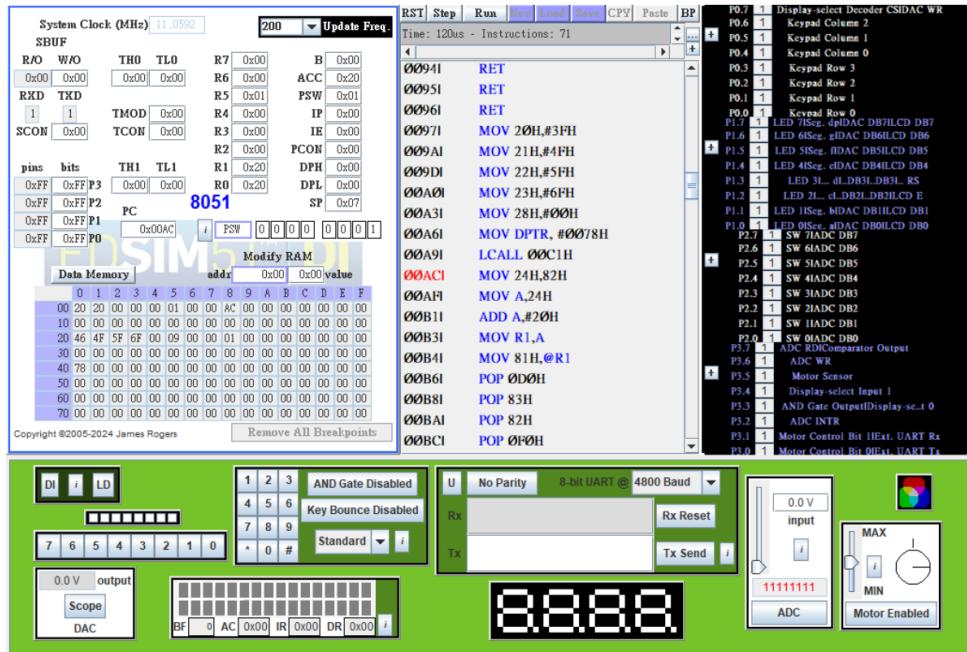
00000009	_Producer	testcoop
00000048	_Consumer	testcoop
00000078	_main	testcoop
00000090	__sdcc_gsinit_startup	testcoop
00000094	__mcs51_genRAMCLEAR	testcoop
00000095	__mcs51_genXINIT	testcoop
00000096	__mcs51_genXRAMCLEAR	testcoop
00000097	_Bootstrap	cooperative
000000C1	_ThreadCreate	cooperative
00000145	_ThreadYield	cooperative
00000196	_ThreadExit	cooperative

2. Screenshot and explanation:

Before **ThreadCreate (main)**:



After **ThreadCreate (main)**:

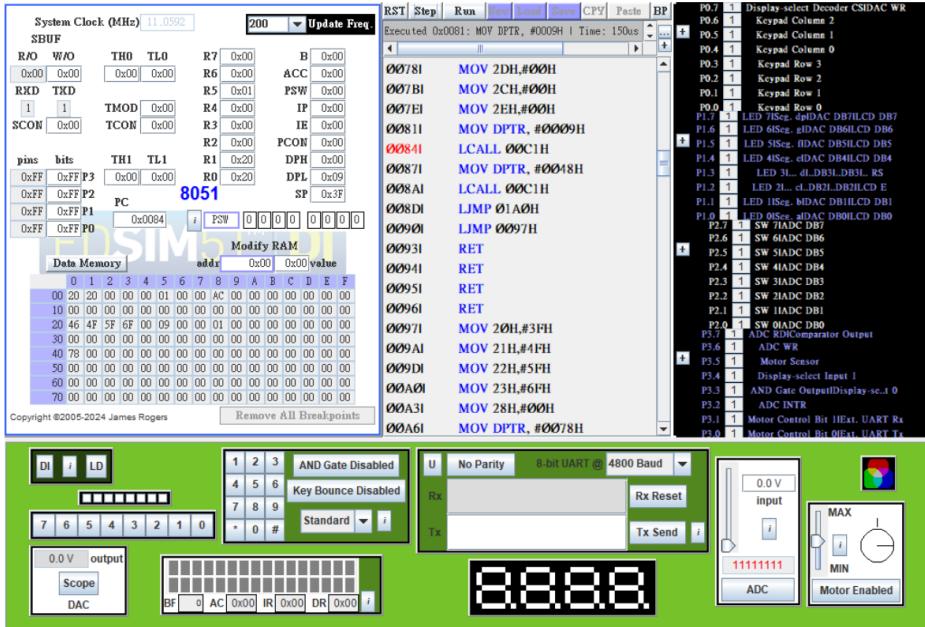


We can see that after **ThreadCreate (main)**, we can see:

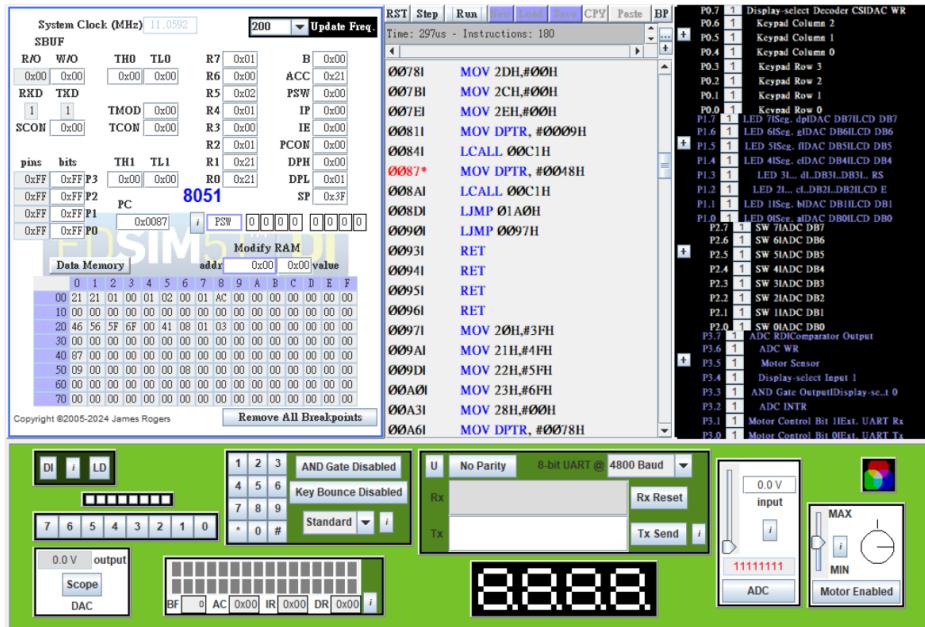
- **SP of thread #0 (address: 0x20) is changed to 46**
- **the bottom of stack of thread #0 (address: 0x40~0x41) is changed to 0078, which is the address of main function**
- **thread bitmap (address: 0x28) is also changed from 0000 to 0001**

meaning main function is assigned to thread #0.

Before ThreadCreate (Producer):



After ThreadCreate (Producer):

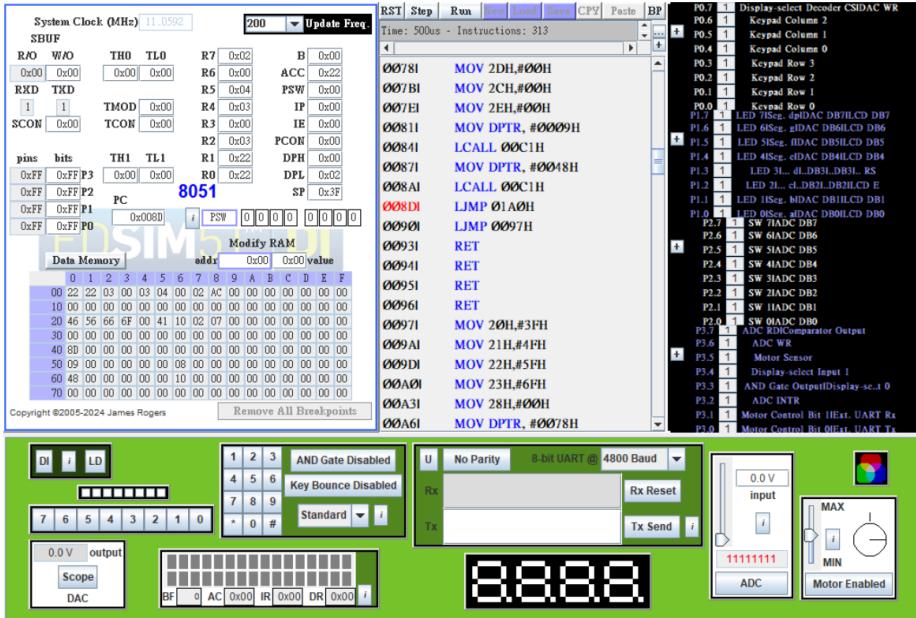


We can see that after **ThreadCreate (Producer)**, we can see:

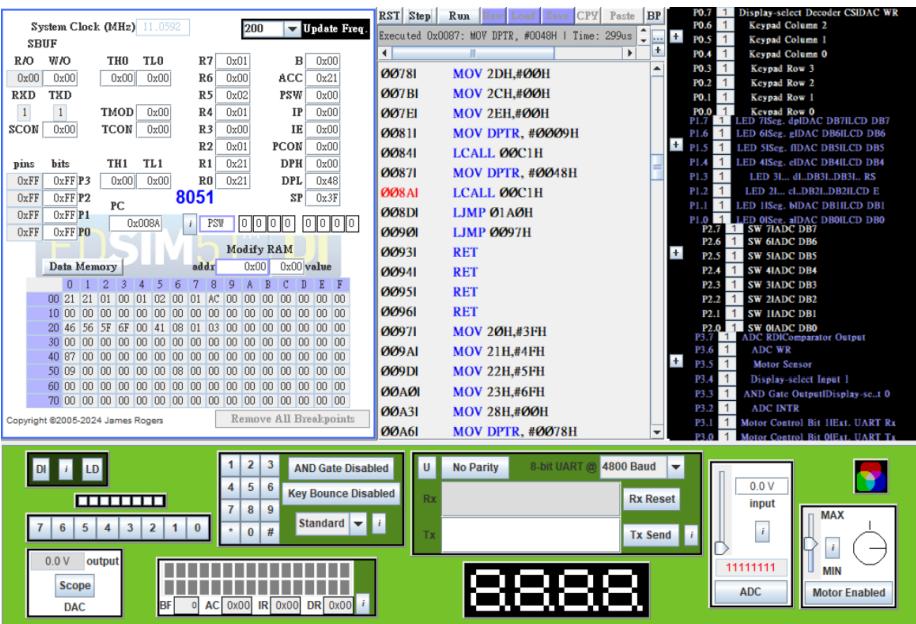
- SP of thread #1 (address: 0x21) is changed to 56**
- the bottom of stack of thread #1 (address: 0x50~0x51) is changed to 0009, which is the address of Producer function**
- thread bitmap (address: 0x28) is also changed from 0001 to 0011**

meaning Producer function is assigned to thread #1.

Before ThreadCreate (Consumer) :



After ThreadCreate (Consumer) :

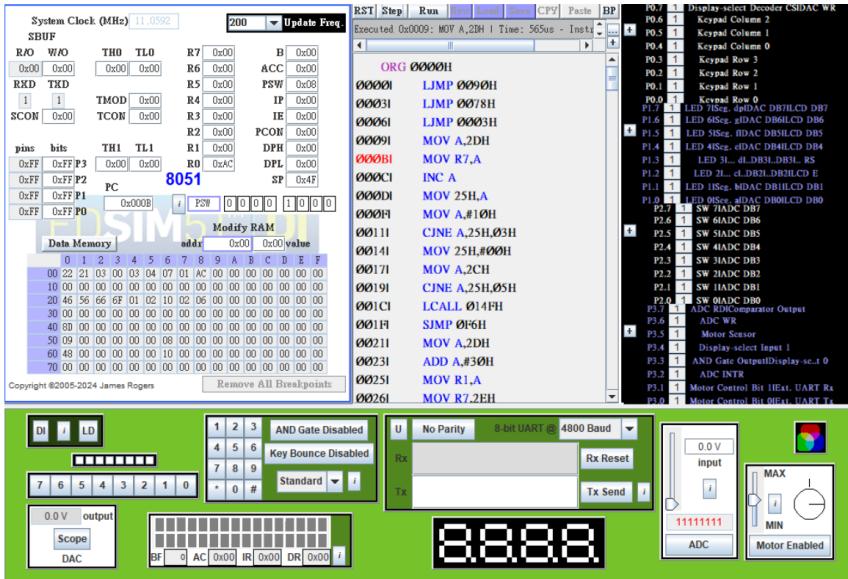


We can see that after **ThreadCreate (Consumer)**, we can see:

- **SP of thread #2 (address: 0x22) is changed to 66**
- **the bottom of stack of thread #2 (address: 0x60~0x61) is changed to 0048, which is the address of Consumer function**
- **thread bitmap (address: 0x28) is also changed from 0011 to 0111**

meaning Consumer function is assigned to thread #2.

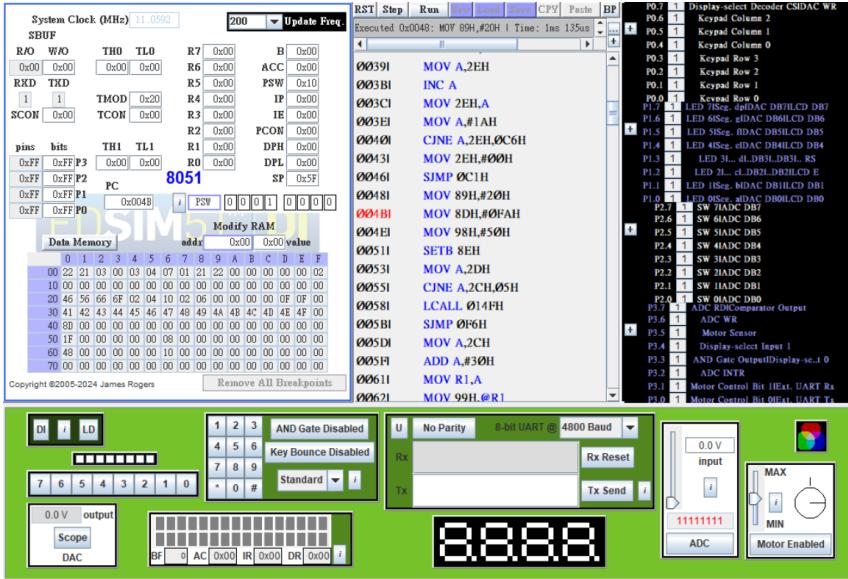
One screenshot when the Producer is running:



How do I know:

- Precise: since PC is in the code section of the function ($0x09 \leq PC < 0x48$).
- Roughly (ignore thread management function): Current thread number (address: 0x24) is 1.

One screenshot when the Consumer is running:



How do I know:

- Precise: since PC is in the code section of the function ($0x48 \leq PC < 0x78$).
- Roughly (ignore thread management function): Current thread number (address: 0x24) is 2.