第10次實習課-電資

學生：林培瑋

2024 Advanced Mixed-Operation System (AMOS) Lab.

期中上機考

2024 Advanced Mixed-Operation System (AMOS) Lab.

評分標準

- 1.(1)(a)(9%)
- 1.(1)(b)(9%)
- 1.(1)(c)(9%)
- 1.(2)(a)(9%)
- 1.(2)(b)(9%)
- 1.(2)(c)(9%)
- 1.(3)(a)(9%)
- 1.(3)(b)(9%)
- 1.(3)(c)(9%)
- 2.(1)(9%)
- 2.(2)(10%)

(1) put necessary Keil Tool DEBUG window screenshots to show your program and execution results including highlighted necessary initial assumptions and subsequent memory, register and stack changes,

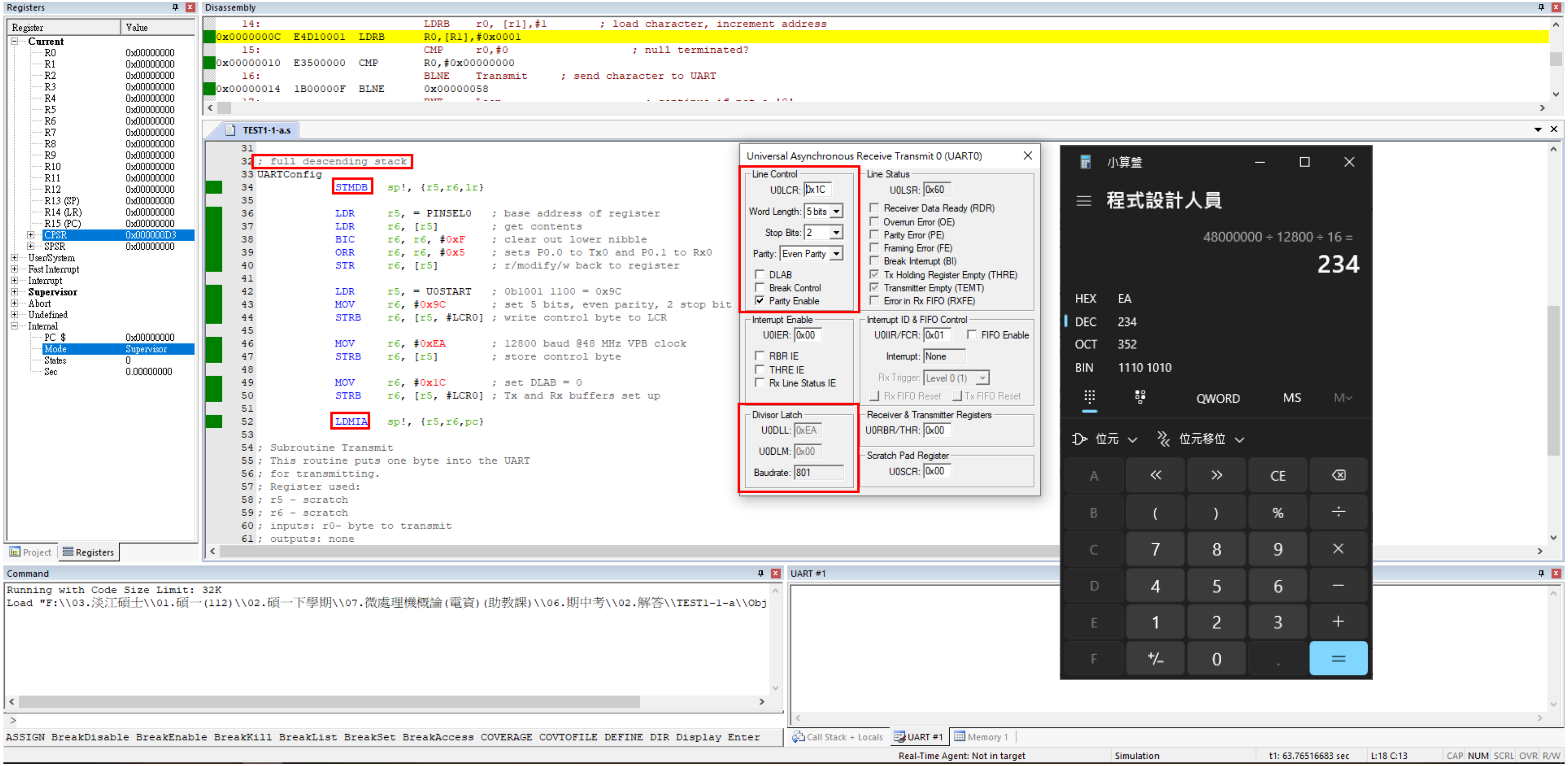
(2) comment student ID+your English name in every screenshots, and

(3) put reports into one word file named by student_ID+your_name.

→補繳分數 = 原始分數*0.8

1. Rewrite the UART Program in Sec. 16.2.5 by using **full descending stack** for subroutine **UARTconfig** and **empty descending stack** for subroutine **Transmit** (both with initial stack pointer 0x40000020, to STM and LDM in the subroutine)

(1) (a) to configure the UART 5 data bits, even parity, 2 stop bits, a Baud rate if the UART is to generate a serial signal at a Baud rate of 12800 Baud using 48 MHz, and show the results in the window of UART0 after execution.



Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

Disassembly

```
    14:                          LDRB     r0, [r1],#1          ; load character, increment address
0x0000000C  E4D10001  LDRB       R0,[R1],#0x0001
    15:                          CMP      r0,#0                        ; null terminated?
0x00000010  E3500000  CMP        R0,#0x00000000
    16:                          BLNE     Transmit     ; send character to UART
0x00000014  1B00000F  BLNE       0x00000058
```

TEST1-1-a.s

```
; full descending stack
UARTConfig
            STMDB   sp!, {r5,r6,lr}

            LDR     r5, = PINSEL0   ; base address of register
            LDR     r6, [r5]        ; get contents
            BIC     r6, r6, #0xF    ; clear out lower nibble
            ORR     r6, r6, #0x5    ; sets P0.0 to Tx0 and P0.1 to Rx0
            STR     r6, [r5]        ; r/modify/w back to register

            LDR     r5, = U0START   ; 0b1001 1100 = 0x9C
            MOV     r6, #0x9C       ; set 5 bits, even parity, 2 stop bit
            STRB    r6, [r5, #LCR0] ; write control byte to LCR

            MOV     r6, #0xEA       ; 12800 baud @48 MHz VPB clock
            STRB    r6, [r5]        ; store control byte

            MOV     r6, #0x1C       ; set DLAB = 0
            STRB    r6, [r5, #LCR0] ; Tx and Rx buffers set up

            LDMIA   sp!, {r5,r6,pc}

; Subroutine Transmit
; This routine puts one byte into the UART
; for transmitting.
; Register used:
; r5 - scratch
; r6 - scratch
; inputs: r0- byte to transmit
; outputs: none
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x1C
Word Length: 5 bits
Stop Bits: 2
Parity: Even Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x60
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
☑ Tx Holding Register Empty (THRE)
☑ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 □ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
U0DLL: 0xEA
U0DLM: 0x00
Baudrate: 801

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

小算盤 — 程式設計人員

48000000 ÷ 12800 ÷ 16 =
234

HEX EA
DEC 234
OCT 352
BIN 1110 1010

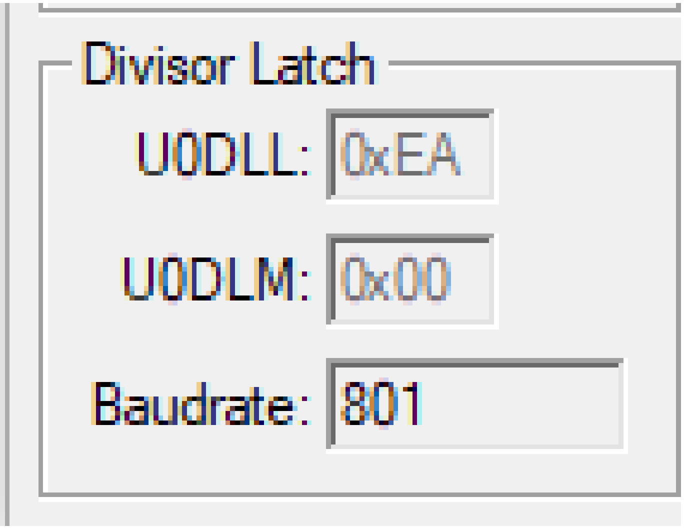
Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-1-a\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

UART #1

Real-Time Agent: Not in target   Simulation   t1: 63.76516683 sec   L:18 C:13

(1) (b) calculate the system clock frequency from the window of UART0 in (a).

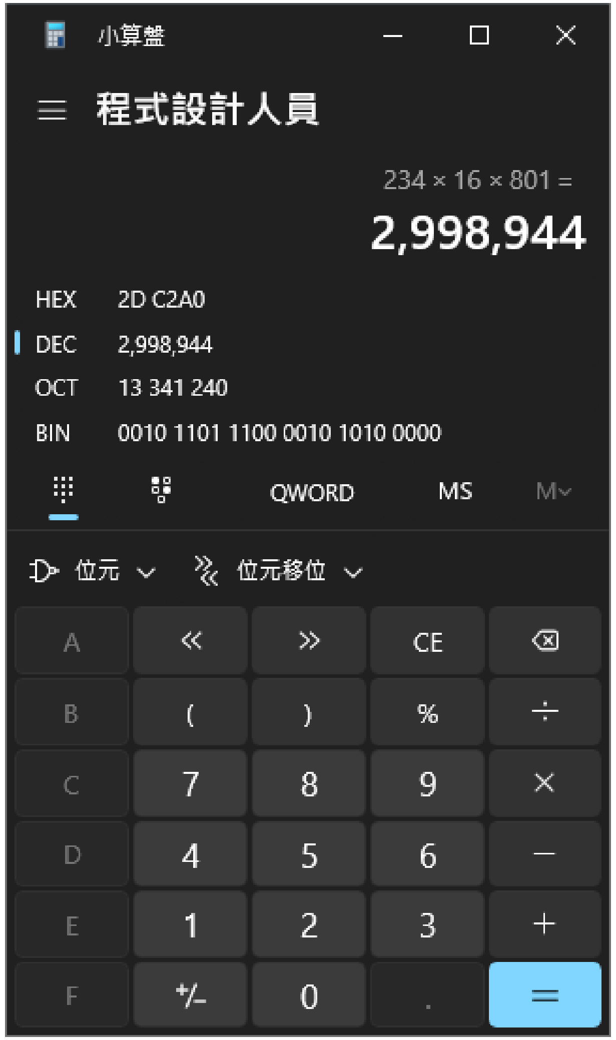


Divisor Latch

U0DLL: 0xEA

U0DLM: 0x00

Baudrate: 801



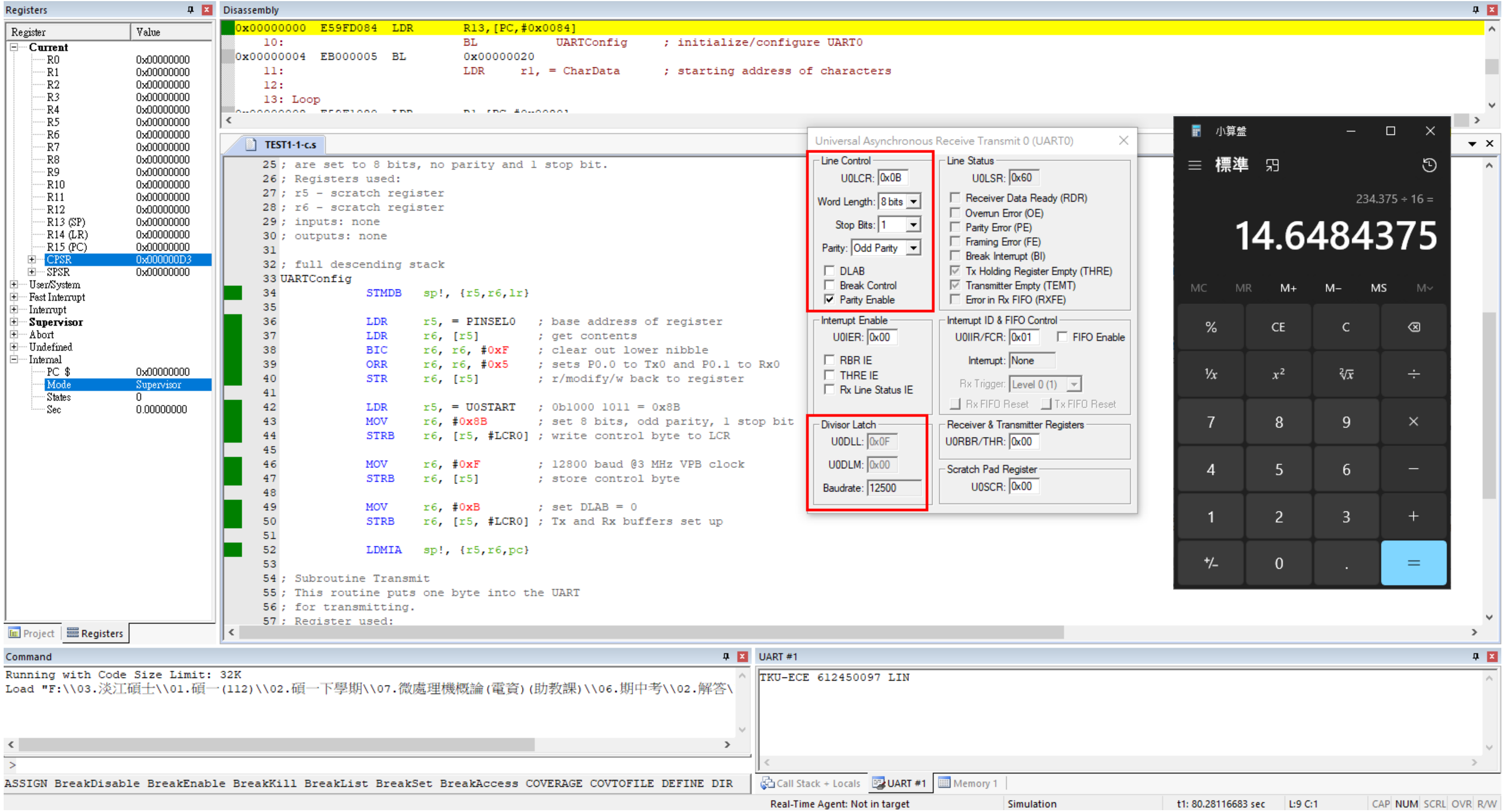
小算盤

程式設計人員

```
234 × 16 × 801 =
2,998,944

HEX   2D C2A0
DEC   2,998,944
OCT   13 341 240
BIN   0010 1101 1100 0010 1010 0000
```

(1) (c) to configure the UART 8 data bits, odd parity, 1 stop bits, a Baud rate if the UART is to generate a serial signal at a Baud rate of 12800 Baud **using Keil Tool LPC 2104 CPU frequency** and show the above results in the window of UART0 after execution.



Registers

```
Register        Value
Current
  R0            0x00000000
  R1            0x00000000
  R2            0x00000000
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x00000000
  R14 (LR)      0x00000000
  R15 (PC)      0x00000000
  CPSR          0x000000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC $          0x00000000
  Mode          Supervisor
  States        0
  Sec           0.00000000
```

Disassembly

```
0x00000000  E59FD084  LDR        R13,[PC,#0x0084]
    10:                          BL          UARTConfig      ; initialize/configure UART0
0x00000004  EB000005  BL         0x00000020
    11:                          LDR     r1, = CharData      ; starting address of characters
    12:
    13: Loop
```

TEST1-1-c.s

```
; are set to 8 bits, no parity and 1 stop bit.
; Registers used:
; r5 - scratch register
; r6 - scratch register
; inputs: none
; outputs: none

; full descending stack
UARTConfig
            STMDB   sp!, {r5,r6,lr}

            LDR     r5, = PINSEL0   ; base address of register
            LDR     r6, [r5]        ; get contents
            BIC     r6, r6, #0xF    ; clear out lower nibble
            ORR     r6, r6, #0x5    ; sets P0.0 to Tx0 and P0.1 to Rx0
            STR     r6, [r5]        ; r/modify/w back to register

            LDR     r5, = U0START   ; 0b1000 1011 = 0x8B
            MOV     r6, #0x8B       ; set 8 bits, odd parity, 1 stop bit
            STRB    r6, [r5, #LCR0] ; write control byte to LCR

            MOV     r6, #0xF        ; 12800 baud @3 MHz VPB clock
            STRB    r6, [r5]        ; store control byte

            MOV     r6, #0xB        ; set DLAB = 0
            STRB    r6, [r5, #LCR0] ; Tx and Rx buffers set up

            LDMIA   sp!, {r5,r6,pc}

; Subroutine Transmit
; This routine puts one byte into the UART
; for transmitting.
; Register used:
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- □ DLAB
- □ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x60
- □ Receiver Data Ready (RDR)
- □ Overrun Error (OE)
- □ Parity Error (PE)
- □ Framing Error (FE)
- □ Break Interrupt (BI)
- ☑ Tx Holding Register Empty (THRE)
- ☑ Transmitter Empty (TEMT)
- □ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- □ RBR IE
- □ THRE IE
- □ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01 □ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- □ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0F
- U0DLM: 0x00
- Baudrate: 12500

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

小算盤 標準

234.375 ÷ 16 =

14.6484375

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR
```

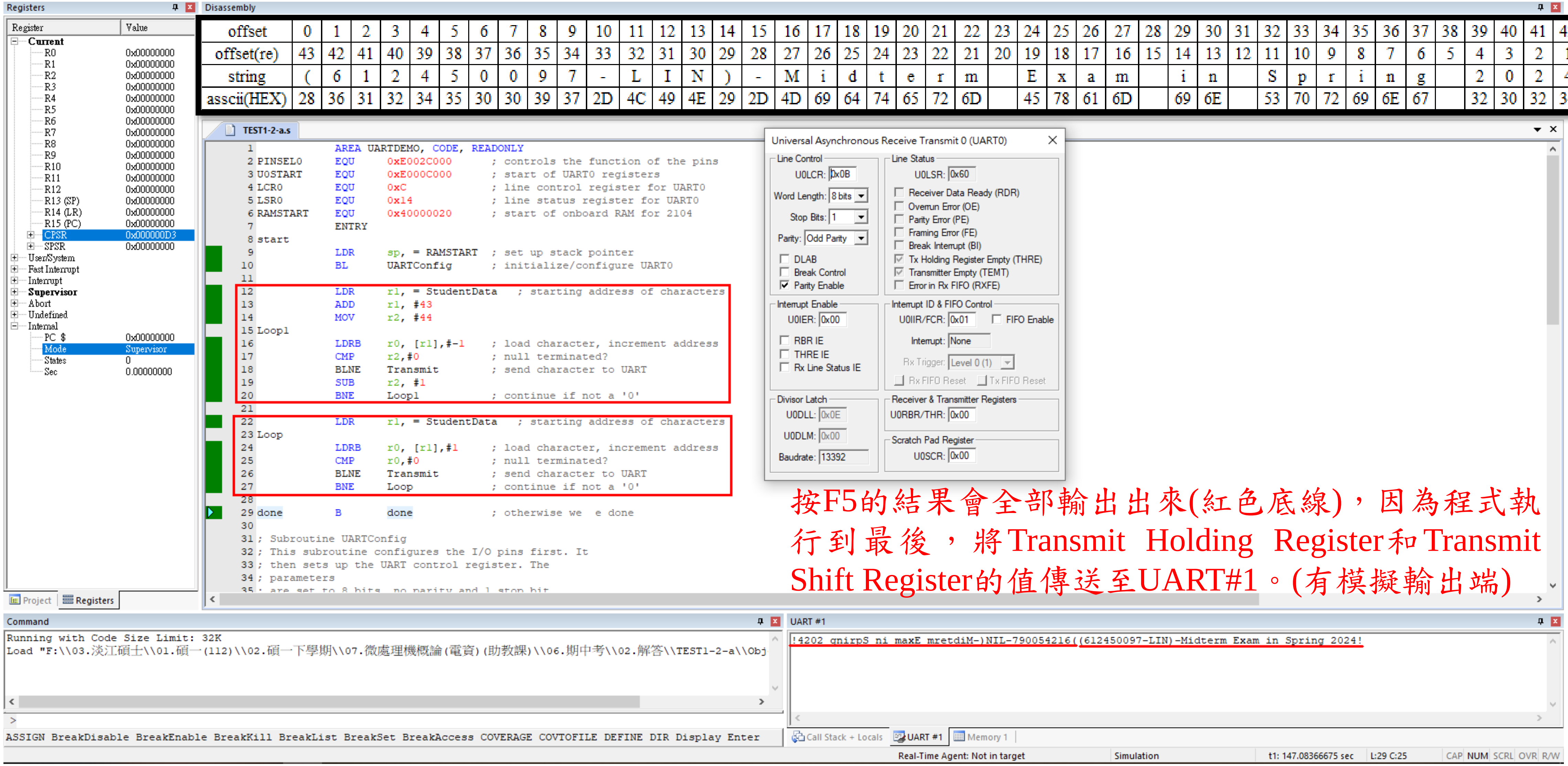
UART #1

```
TKU-ECE 612450097 LIN
```

Real-Time Agent: Not in target | Simulation | t1: 80.28116683 sec | L:9 C:1

(2) to include the declaration of the string "**(ID-Name)-Midterm Exam in Spring 2024!**" as variable **StudentData**. Use calls to subroutine **Transmit** to do the following 3 steps

(a) display **reversely the string** and **continuously the string (F5)**



offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21

```
Register          Value
Current
  R0              0x00000000
  R1              0x00000000
  R2              0x00000000
  R3              0x00000000
  R4              0x00000000
  R5              0x00000000
  R6              0x00000000
  R7              0x00000000
  R8              0x00000000
  R9              0x00000000
  R10             0x00000000
  R11             0x00000000
  R12             0x00000000
  R13 (SP)        0x00000000
  R14 (LR)        0x00000000
  R15 (PC)        0x00000000
  CPSR            0x000000D3
  SPSR            0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC  $           0x00000000
  Mode            Supervisor
  States          0
  Sec             0.00000000
```

```
            AREA UARTDEMO, CODE, READONLY
PINSEL0     EQU     0xE002C000        ; controls the function of the pins
U0START     EQU     0xE000C000        ; start of UART0 registers
LCR0        EQU     0xC               ; line control register for UART0
LSR0        EQU     0x14              ; line status register for UART0
RAMSTART    EQU     0x40000020        ; start of onboard RAM for 2104
            ENTRY
start
            LDR     sp, = RAMSTART    ; set up stack pointer
            BL      UARTConfig        ; initialize/configure UART0

            LDR     r1, = StudentData   ; starting address of characters
            ADD     r1, #43
            MOV     r2, #44
Loop1
            LDRB    r0, [r1],#-1      ; load character, increment address
            CMP     r2,#0             ; null terminated?
            BLNE    Transmit          ; send character to UART
            SUB     r2, #1
            BNE     Loop1             ; continue if not a '0'

            LDR     r1, = StudentData   ; starting address of characters
Loop
            LDRB    r0, [r1],#1       ; load character, increment address
            CMP     r0,#0             ; null terminated?
            BLNE    Transmit          ; send character to UART
            BNE     Loop              ; continue if not a '0'

done        B       done              ; otherwise we  e done

; Subroutine UARTConfig
; This subroutine configures the I/O pins first. It
; then sets up the UART control register. The
; parameters
; are set to 8 bits, no parity and 1 stop bit
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- ☐ DLAB
- ☐ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x60
- ☐ Receiver Data Ready (RDR)
- ☐ Overrun Error (OE)
- ☐ Parity Error (PE)
- ☐ Framing Error (FE)
- ☐ Break Interrupt (BI)
- ☑ Tx Holding Register Empty (THRE)
- ☑ Transmitter Empty (TEMT)
- ☐ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- ☐ RBR IE
- ☐ THRE IE
- ☐ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01 ☐ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- ☐ Rx FIFO Reset ☐ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0E
- U0DLM: 0x00
- Baudrate: 13392

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

按F5的結果會全部輸出出來(紅色底線)，因為程式執行到最後，將Transmit Holding Register和Transmit Shift Register的值傳送至UART#1。(有模擬輸出端)

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-a\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

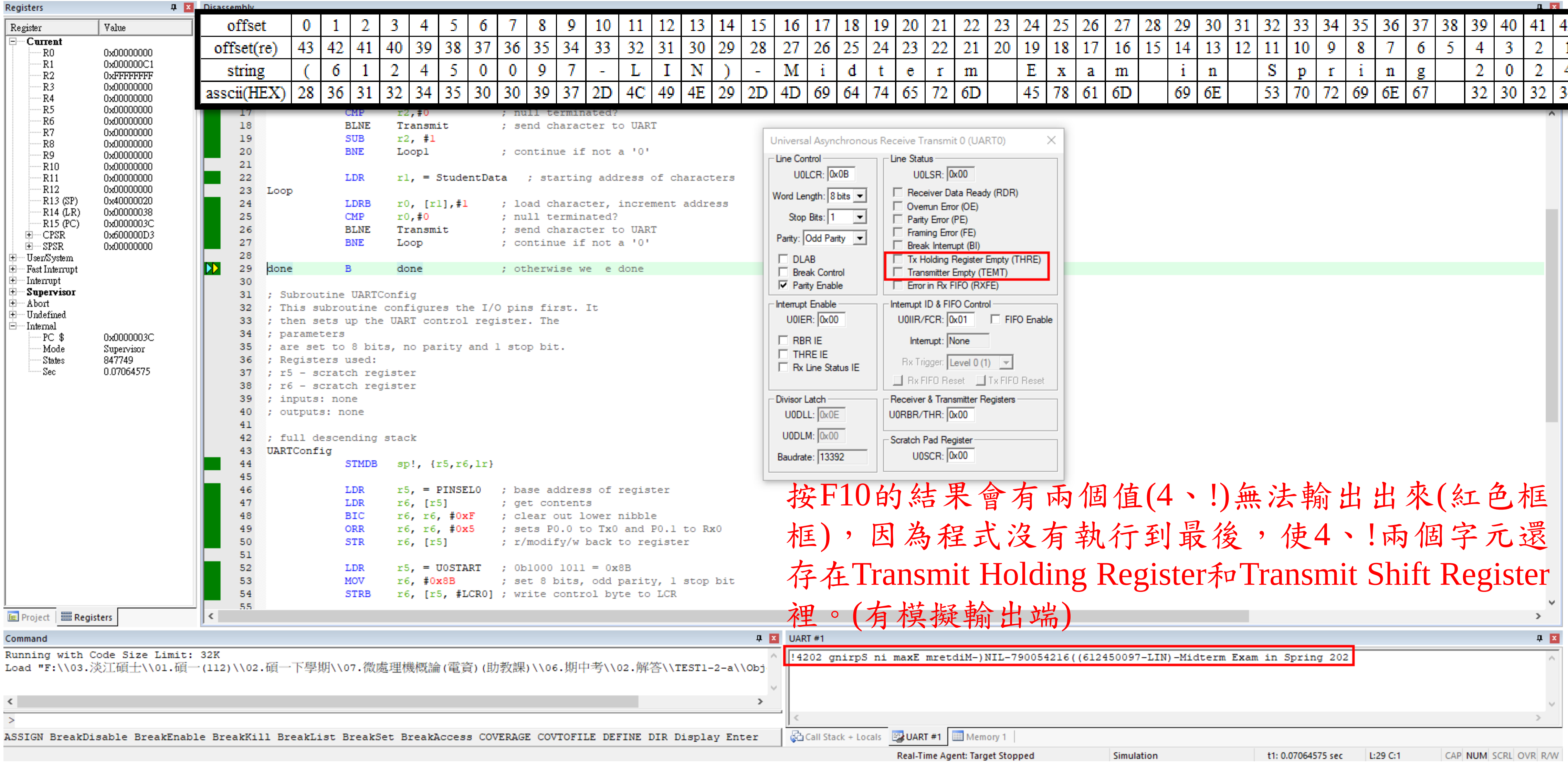
```
!4202 gnirpS ni maxE mretdiM-)NIL-790054216((612450097-LIN)-Midterm Exam in Spring 2024!
```

```
Real-Time Agent: Not in target    Simulation    t1: 147.08366675 sec    L:29 C:25    CAP NUM SCRL OVR R/W
```

(2) to include the declaration of the string **“(ID-Name)-Midterm Exam in Spring 2024!”** as variable **StudentData**. Use calls to subroutine **Transmit** to do the following 3 steps

(a) display **reversely the string** and **continuously the string (F10)**

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

Register	Value
Current	
R0	0x00000000
R1	0x000000C1
R2	0xFFFFFFFF
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x40000020
R14 (LR)	0x00000038
R15 (PC)	0x0000003C
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x0000003C
Mode	Supervisor
States	847749
Sec	0.07064575

Disassembly

```
                CMP     r2,#0           ; null terminated?
                BLNE    Transmit        ; send character to UART
                SUB     r2, #1
                BNE     Loop1           ; continue if not a '0'

                LDR     r1, = StudentData   ; starting address of characters
Loop
                LDRB    r0, [r1],#1     ; load character, increment address
                CMP     r0,#0           ; null terminated?
                BLNE    Transmit        ; send character to UART
                BNE     Loop            ; continue if not a '0'

done            B       done            ; otherwise we  e done

; Subroutine UARTConfig
; This subroutine configures the I/O pins first. It
; then sets up the UART control register. The
; parameters
; are set to 8 bits, no parity and 1 stop bit.
; Registers used:
; r5 - scratch register
; r6 - scratch register
; inputs: none
; outputs: none

; full descending stack
UARTConfig
                STMDB   sp!, {r5,r6,lr}

                LDR     r5, = PINSEL0   ; base address of register
                LDR     r6, [r5]        ; get contents
                BIC     r6, r6, #0xF    ; clear out lower nibble
                ORR     r6, r6, #0x5    ; sets P0.0 to Tx0 and P0.1 to Rx0
                STR     r6, [r5]        ; r/modify/w back to register

                LDR     r5, = U0START   ; 0b1000 1011 = 0x8B
                MOV     r6, #0x8B       ; set 8 bits, odd parity, 1 stop bit
                STRB    r6, [r5, #LCR0] ; write control byte to LCR
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
☐ DLAB
☐ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x00
☐ Receiver Data Ready (RDR)
☐ Overrun Error (OE)
☐ Parity Error (PE)
☐ Framing Error (FE)
☐ Break Interrupt (BI)
☐ Tx Holding Register Empty (THRE)
☐ Transmitter Empty (TEMT)
☐ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
☐ RBR IE
☐ THRE IE
☐ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 ☐ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
☐ Rx FIFO Reset ☐ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

按F10的結果會有兩個值(4、!)無法輸出出來(紅色框框)，因為程式沒有執行到最後，使4、!兩個字元還存在Transmit Holding Register和Transmit Shift Register裡。(有模擬輸出端)

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-a\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

UART #1

```
!4202 gnirpS ni maxE mretdiM-)NIL-790054216((612450097-LIN)-Midterm Exam in Spring 202
```

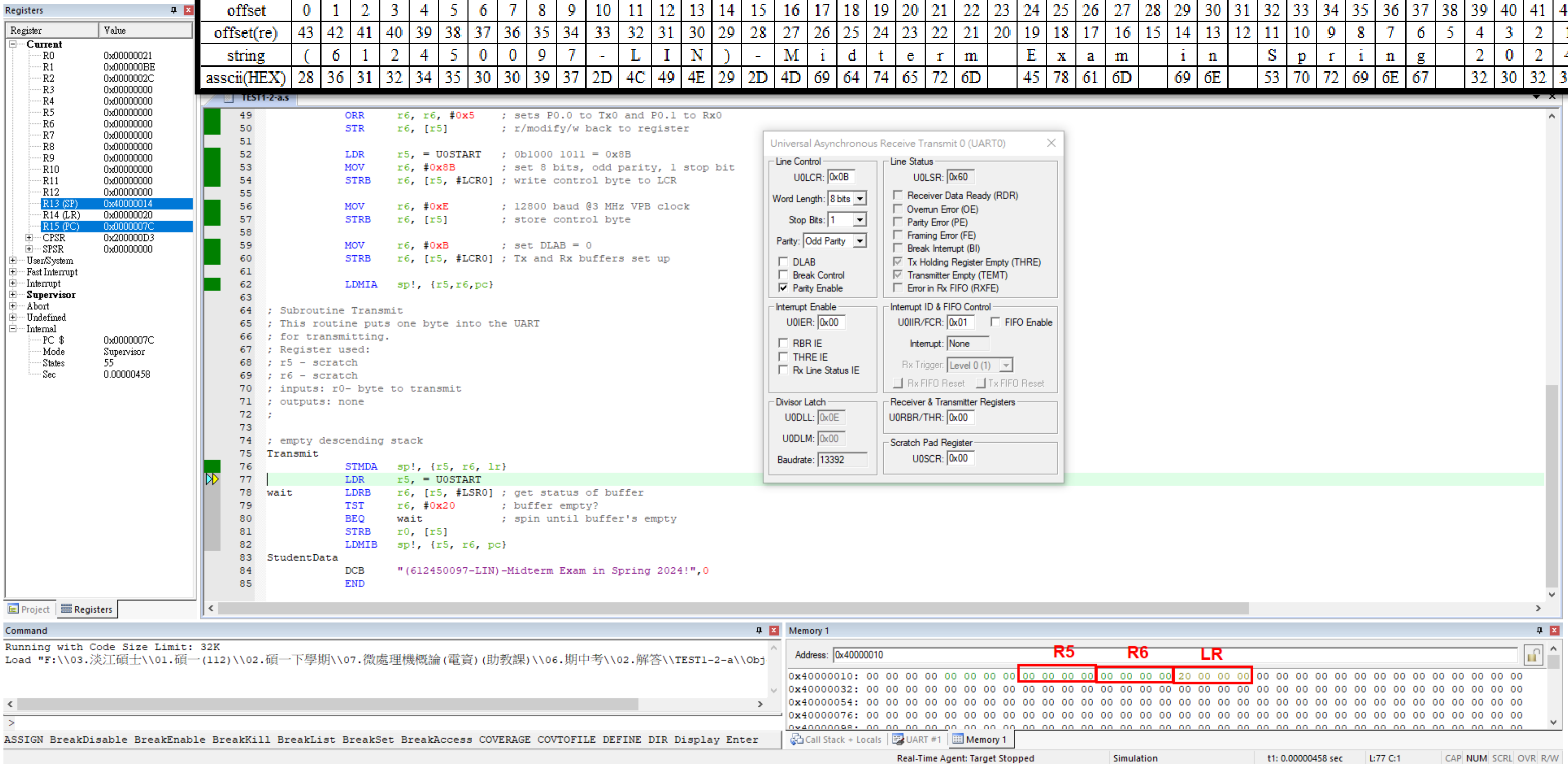
(2) to include the declaration of the string **"(ID-Name)-Midterm Exam in Spring 2024!"** as variable **StudentData**. Use calls to subroutine **Transmit** to do the following 3 steps

(a) display **reversely the string** and **continuously the string (F11)**

(highlight the stack elements with the related registers stored for subroutines Receive and Transmit)

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72

offset	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

```
Register        Value
Current
  R0            0x00000021
  R1            0x000000BE
  R2            0x0000002C
  R3            0x00000000
  R4            0x00000000
  R5            0x00000000
  R6            0x00000000
  R7            0x00000000
  R8            0x00000000
  R9            0x00000000
  R10           0x00000000
  R11           0x00000000
  R12           0x00000000
  R13 (SP)      0x40000014
  R14 (LR)      0x00000020
  R15 (PC)      0x0000007C
  CPSR          0x200000D3
  SPSR          0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC $          0x0000007C
  Mode          Supervisor
  States        55
  Sec           0.00000458
```

TEST1-2-a.s

```
            ORR     r6, r6, #0x5     ; sets P0.0 to Tx0 and P0.1 to Rx0
            STR     r6, [r5]         ; r/modify/w back to register

            LDR     r5, = U0START    ; 0b1000 1011 = 0x8B
            MOV     r6, #0x8B        ; set 8 bits, odd parity, 1 stop bit
            STRB    r6, [r5, #LCR0] ; write control byte to LCR

            MOV     r6, #0xE         ; 12800 baud @3 MHz VPB clock
            STRB    r6, [r5]         ; store control byte

            MOV     r6, #0xB         ; set DLAB = 0
            STRB    r6, [r5, #LCR0] ; Tx and Rx buffers set up

            LDMIA   sp!, {r5,r6,pc}

; Subroutine Transmit
; This routine puts one byte into the UART
; for transmitting.
; Register used:
; r5 - scratch
; r6 - scratch
; inputs: r0- byte to transmit
; outputs: none
;

; empty descending stack
Transmit
            STMDA   sp!, {r5, r6, lr}
            LDR     r5, = U0START
wait        LDRB    r6, [r5, #LSR0] ; get status of buffer
            TST     r6, #0x20        ; buffer empty?
            BEQ     wait             ; spin until buffer's empty
            STRB    r0, [r5]
            LDMIB   sp!, {r5, r6, pc}
StudentData
            DCB     "(612450097-LIN)-Midterm Exam in Spring 2024!",0
            END
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- □ DLAB
- □ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x60
- □ Receiver Data Ready (RDR)
- □ Overrun Error (OE)
- □ Parity Error (PE)
- □ Framing Error (FE)
- □ Break Interrupt (BI)
- ☑ Tx Holding Register Empty (THRE)
- ☑ Transmitter Empty (TEMT)
- □ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- □ RBR IE
- □ THRE IE
- □ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01 □ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- □ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0E
- U0DLM: 0x00
- Baudrate: 13392

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-a\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 1

Address: 0x40000010

R5 R6 LR

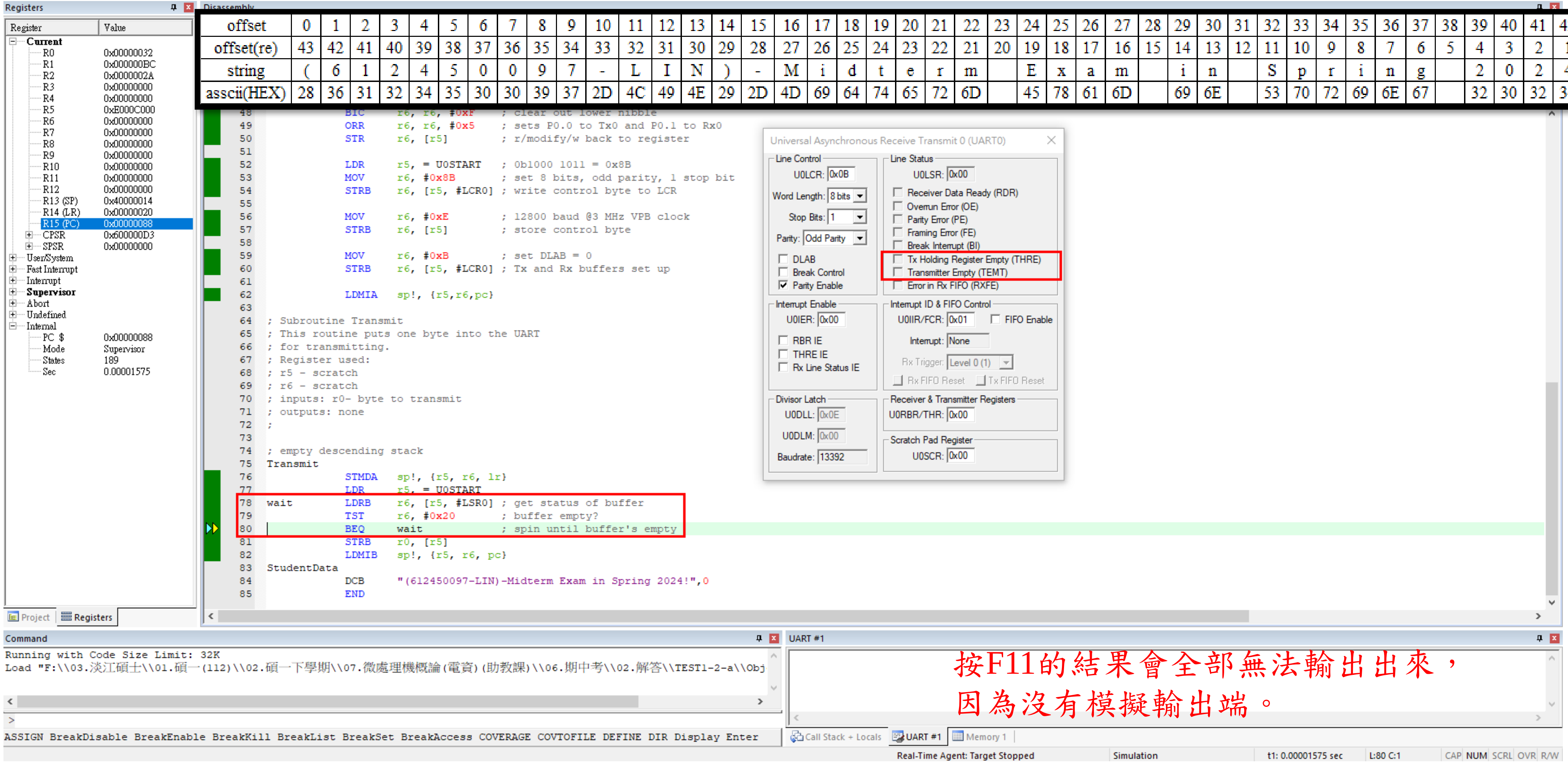
```
0x40000010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000032: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000054: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40000076: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Real-Time Agent: Target Stopped   Simulation   t1: 0.00000458 sec   L:77 C:1

(2) to include the declaration of the string “**(ID-Name)-Midterm Exam in Spring 2024!**” as variable **StudentData**. Use calls to subroutine **Transmit** to do the following 3 steps

(a) display **reversely the string** and **continuously the string (F11)**

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

Register	Value
Current	
R0	0x00000032
R1	0x000000BC
R2	0x0000002A
R3	0x00000000
R4	0x00000000
R5	0xE000C000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x40000014
R14 (LR)	0x00000020
R15 (PC)	0x00000088
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000088
Mode	Supervisor
States	189
Sec	0.00001575

Disassembly

```
                BIC     r6, r6, #0xF      ; clear out lower nibble
                ORR     r6, r6, #0x5      ; sets P0.0 to Tx0 and P0.1 to Rx0
                STR     r6, [r5]          ; r/modify/w back to register

                LDR     r5, = U0START     ; 0b1000 1011 = 0x8B
                MOV     r6, #0x8B         ; set 8 bits, odd parity, 1 stop bit
                STRB    r6, [r5, #LCR0] ; write control byte to LCR

                MOV     r6, #0xE          ; 12800 baud @3 MHz VPB clock
                STRB    r6, [r5]          ; store control byte

                MOV     r6, #0xB          ; set DLAB = 0
                STRB    r6, [r5, #LCR0] ; Tx and Rx buffers set up

                LDMIA   sp!, {r5,r6,pc}

; Subroutine Transmit
; This routine puts one byte into the UART
; for transmitting.
; Register used:
; r5 - scratch
; r6 - scratch
; inputs: r0- byte to transmit
; outputs: none
;

; empty descending stack
Transmit
                STMDA   sp!, {r5, r6, lr}
                LDR     r5, = U0START
wait            LDRB    r6, [r5, #LSR0] ; get status of buffer
                TST     r6, #0x20         ; buffer empty?
                BEQ     wait              ; spin until buffer's empty
                STRB    r0, [r5]
                LDMIB   sp!, {r5, r6, pc}
StudentData
                DCB     "(612450097-LIN)-Midterm Exam in Spring 2024!",0
                END
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x00
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
□ Tx Holding Register Empty (THRE)
□ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 □ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

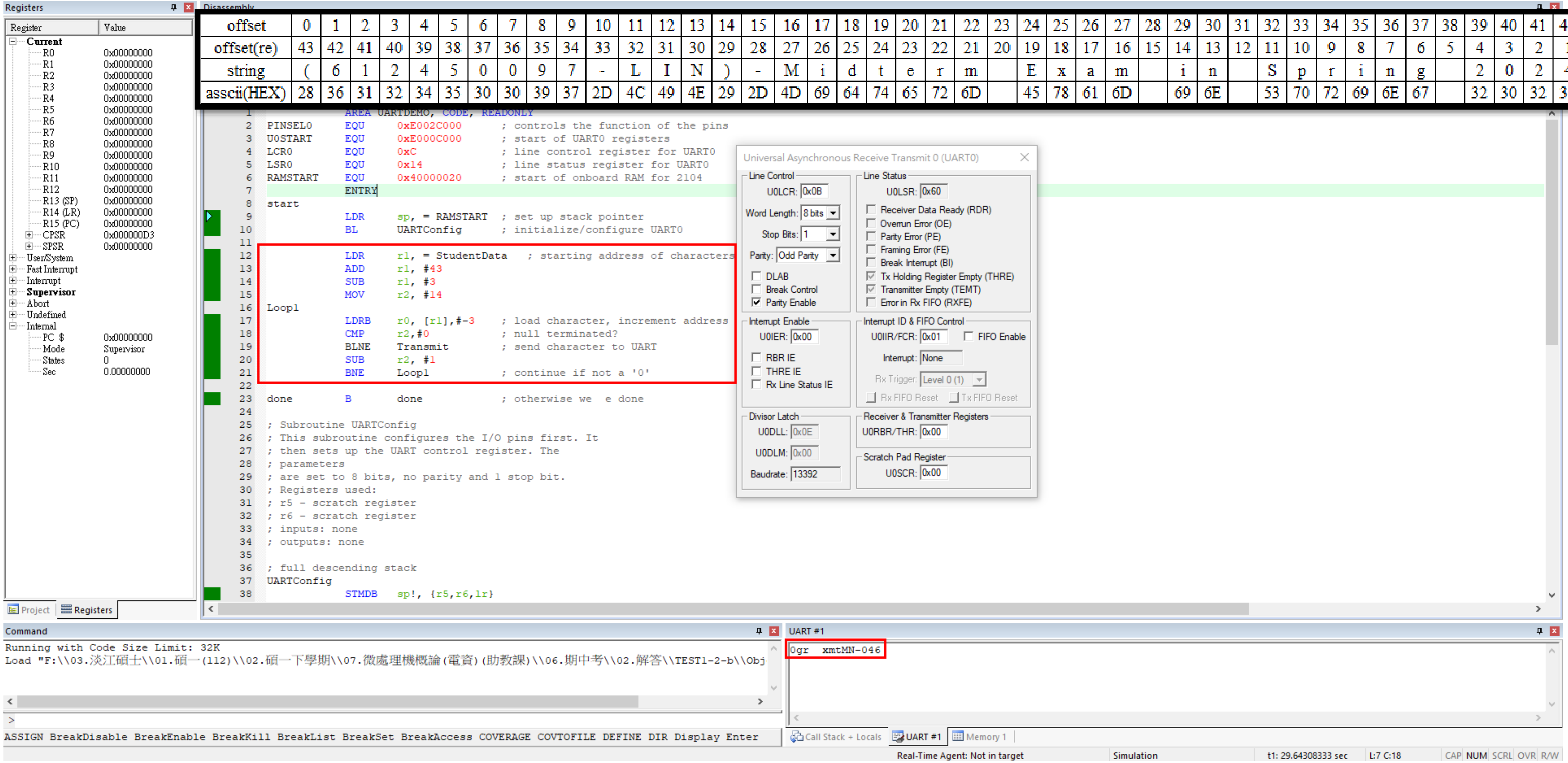
Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-a\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

UART #1

按F11的結果會全部無法輸出出來，
因為沒有模擬輸出端。

(2) (b) display **reversely the string characters in multiple-of-3 positions** (assuming the last character starts from position 0) (F5)



offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21

```
                AREA UARTDEMO, CODE, READONLY
PINSEL0         EQU     0xE002C000      ; controls the function of the pins
U0START         EQU     0xE000C000      ; start of UART0 registers
LCR0            EQU     0xC             ; line control register for UART0
LSR0            EQU     0x14            ; line status register for UART0
RAMSTART        EQU     0x40000020      ; start of onboard RAM for 2104
                ENTRY
start
                LDR     sp, = RAMSTART  ; set up stack pointer
                BL      UARTConfig      ; initialize/configure UART0

                LDR     r1, = StudentData   ; starting address of characters
                ADD     r1, #43
                SUB     r1, #3
                MOV     r2, #14
Loop1
                LDRB    r0, [r1],#-3    ; load character, increment address
                CMP     r2,#0           ; null terminated?
                BLNE    Transmit        ; send character to UART
                SUB     r2, #1
                BNE     Loop1           ; continue if not a '0'

done            B       done            ; otherwise we  e done

; Subroutine UARTConfig
; This subroutine configures the I/O pins first. It
; then sets up the UART control register. The
; parameters
; are set to 8 bits, no parity and 1 stop bit.
; Registers used:
; r5 - scratch register
; r6 - scratch register
; inputs: none
; outputs: none

; full descending stack
UARTConfig
                STMDB   sp!, {r5,r6,lr}
```

Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x60
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
☑ Tx Holding Register Empty (THRE)
☑ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 □ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

Command

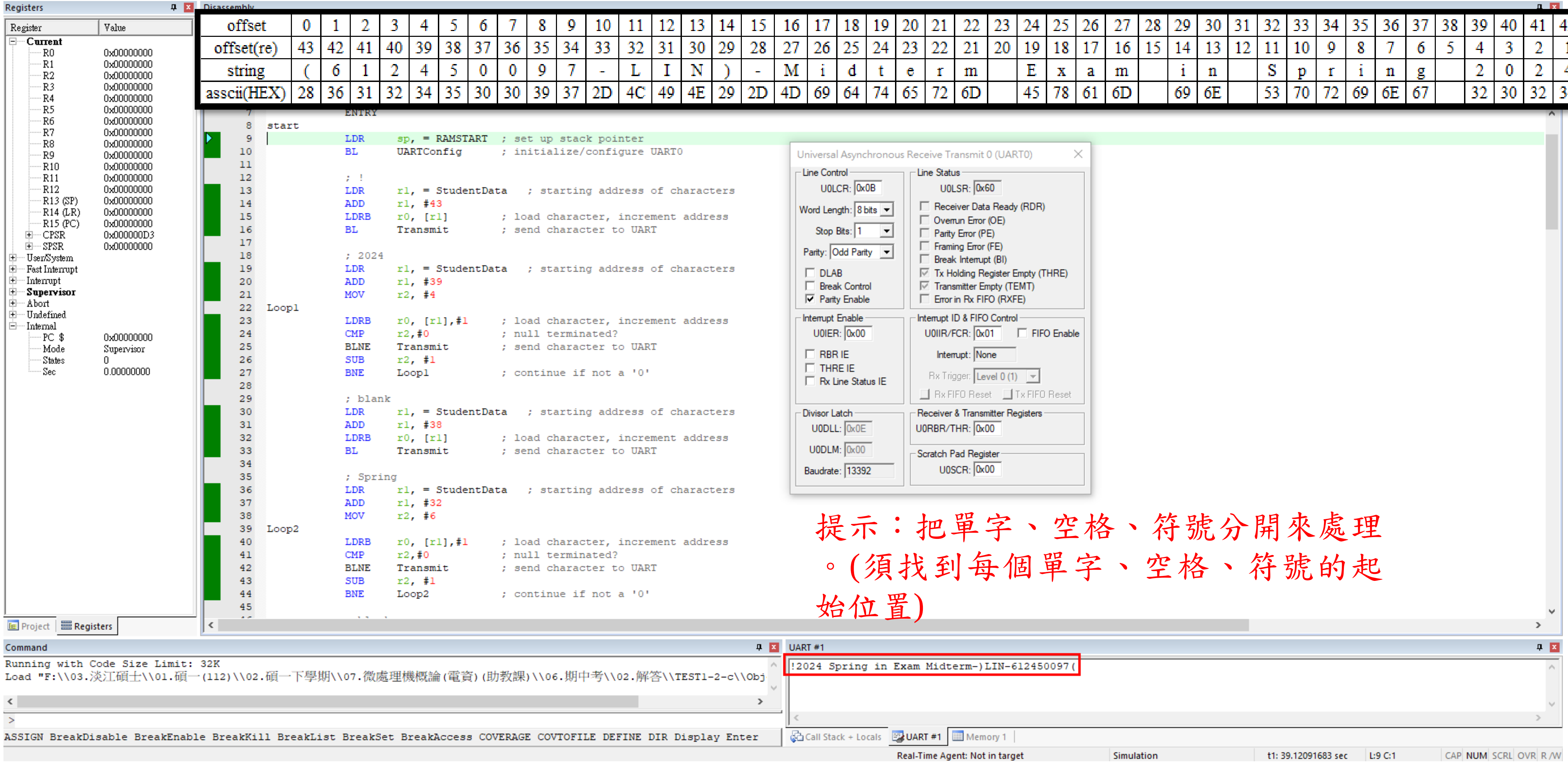
```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-b\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

UART #1

```
0gr  xmtMN-046
```

(2) (c) display **reversely the string "words"** (F5)

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

Disassembly

```
                ENTRY
start
                LDR     sp, = RAMSTART  ; set up stack pointer
                BL      UARTConfig      ; initialize/configure UART0

                ; !
                LDR     r1, = StudentData   ; starting address of characters
                ADD     r1, #43
                LDRB    r0, [r1]        ; load character, increment address
                BL      Transmit        ; send character to UART

                ; 2024
                LDR     r1, = StudentData   ; starting address of characters
                ADD     r1, #39
                MOV     r2, #4
Loop1
                LDRB    r0, [r1],#1     ; load character, increment address
                CMP     r2,#0           ; null terminated?
                BLNE    Transmit        ; send character to UART
                SUB     r2, #1
                BNE     Loop1           ; continue if not a '0'

                ; blank
                LDR     r1, = StudentData   ; starting address of characters
                ADD     r1, #38
                LDRB    r0, [r1]        ; load character, increment address
                BL      Transmit        ; send character to UART

                ; Spring
                LDR     r1, = StudentData   ; starting address of characters
                ADD     r1, #32
                MOV     r2, #6
Loop2
                LDRB    r0, [r1],#1     ; load character, increment address
                CMP     r2,#0           ; null terminated?
                BLNE    Transmit        ; send character to UART
                SUB     r2, #1
                BNE     Loop2           ; continue if not a '0'
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x60
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
☑ Tx Holding Register Empty (THRE)
☑ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 □ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

提示：把單字、空格、符號分開來處理。(須找到每個單字、空格、符號的起始位置)

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-2-c\\Obj
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

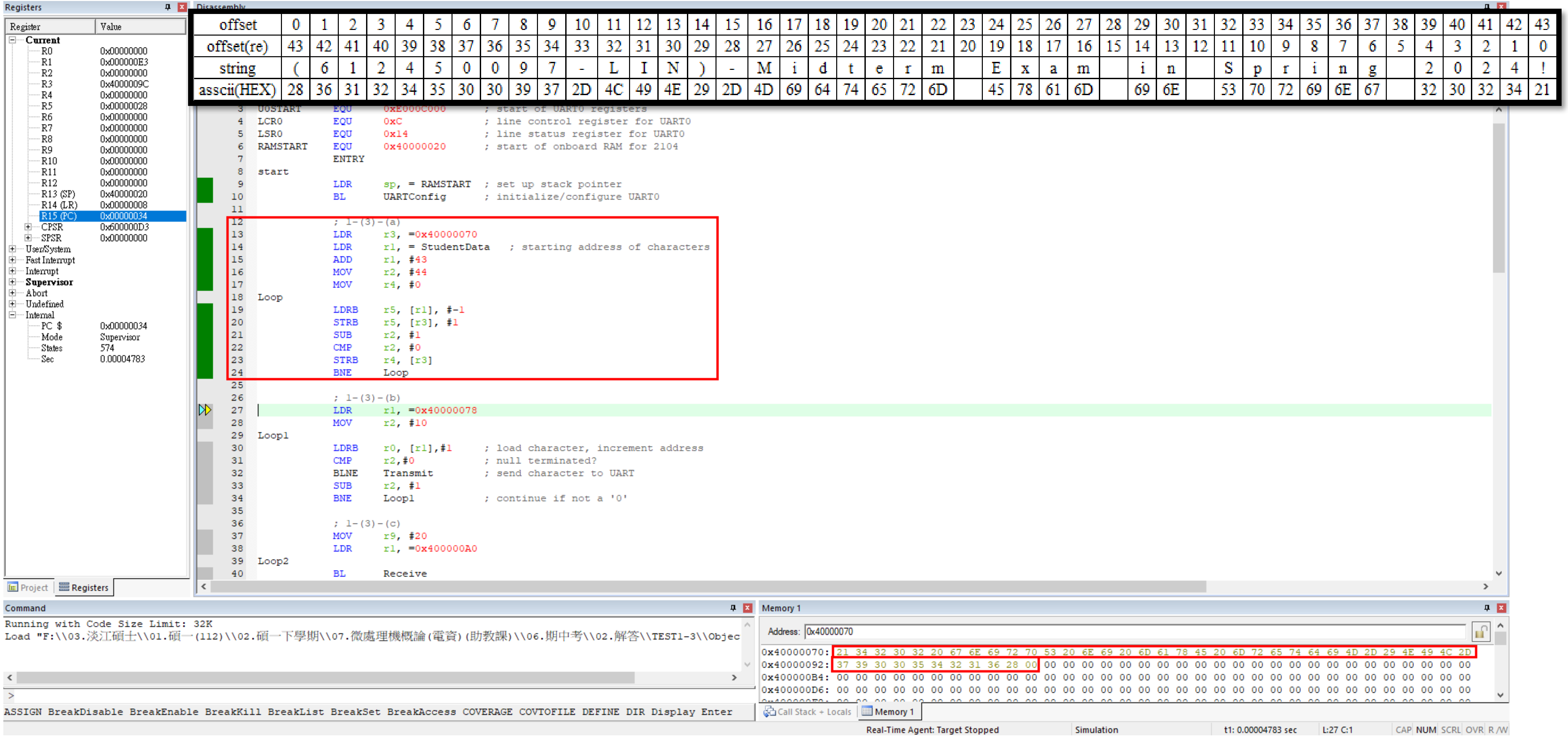
UART #1

```
!2024 Spring in Exam Midterm-)LIN-612450097(
```

Real-Time Agent: Not in target   Simulation   t1: 39.12091683 sec   L:9 C:1   CAP NUM SCRL OVR R/W

(3) to include subroutines **Receive** (using **empty ascending stack** with initial stack pointer 0x40000020, to STM and LDM in the subroutine) to receive an **error-free** byte data from the receiver buffer register to R1.

(a) to copy the string (variable **StudentData**) **reversely** to memory starting from address 0x4000007**0**.



offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21

Registers

Register	Value
Current	
R0	0x00000000
R1	0x000000E3
R2	0x00000000
R3	0x4000009C
R4	0x00000000
R5	0x00000028
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x40000020
R14 (LR)	0x00000008
R15 (PC)	0x00000034
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000034
Mode	Supervisor
States	574
Sec	0.00004783

```
U0START     EQU     0xE000C000       ; start of UART0 registers
LCR0        EQU     0xC              ; line control register for UART0
LSR0        EQU     0x14             ; line status register for UART0
RAMSTART    EQU     0x40000020       ; start of onboard RAM for 2104
            ENTRY
start
            LDR     sp, = RAMSTART   ; set up stack pointer
            BL      UARTConfig       ; initialize/configure UART0

            ; 1-(3)-(a)
            LDR     r3, =0x40000070
            LDR     r1, = StudentData   ; starting address of characters
            ADD     r1, #43
            MOV     r2, #44
            MOV     r4, #0
Loop
            LDRB    r5, [r1], #-1
            STRB    r5, [r3], #1
            SUB     r2, #1
            CMP     r2, #0
            STRB    r4, [r3]
            BNE     Loop

            ; 1-(3)-(b)
            LDR     r1, =0x40000078
            MOV     r2, #10
Loop1
            LDRB    r0, [r1],#1      ; load character, increment address
            CMP     r2,#0            ; null terminated?
            BLNE    Transmit         ; send character to UART
            SUB     r2, #1
            BNE     Loop1            ; continue if not a '0'

            ; 1-(3)-(c)
            MOV     r9, #20
            LDR     r1, =0x400000A0
Loop2
            BL      Receive
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-3\\Objec
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 1

Address: 0x40000070

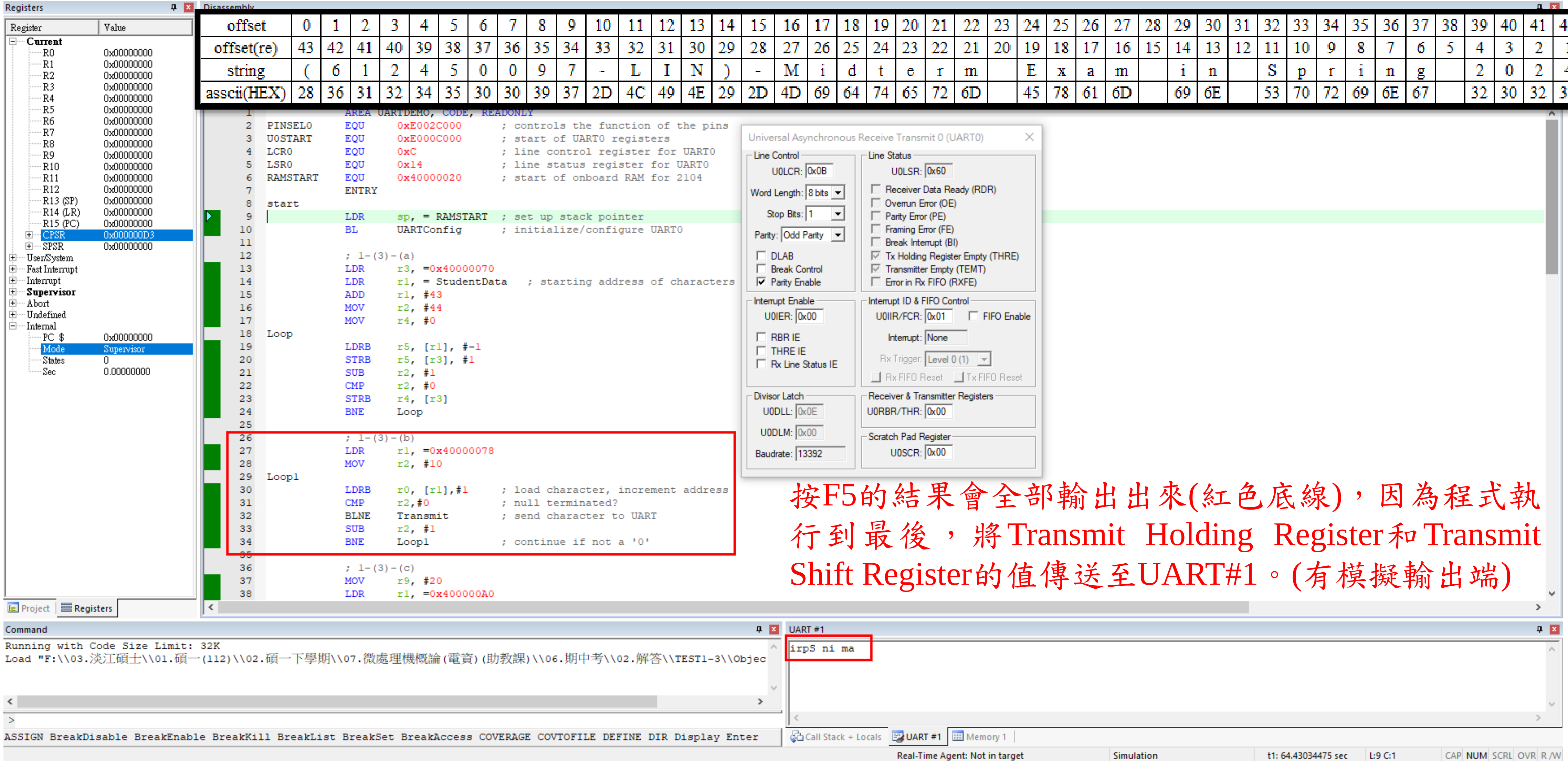
```
0x40000070: 21 34 32 30 32 20 67 6E 69 72 70 53 20 6E 69 20 6D 61 78 45 20 6D 72 65 74 64 69 4D 2D 29 4E 49 4C 2D
0x40000092: 37 39 30 30 35 34 32 31 36 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000B4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400000D6: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Real-Time Agent: Target Stopped   Simulation   t1: 0.00004783 sec   L:27 C:1   CAP NUM SCRL OVR R /W

(3) (b) to use calls to subroutine **Transmit** to display a sequence of **10** characters at memory address 0x400000**78** in the **UART #1** window after program execution by using **F5 (Run)**.

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72

offset	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Register	Value
Current	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000000
CPSR	0x000000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000000
Mode	Supervisor
States	0
Sec	0.00000000

```
            AREA UARTDEMO, CODE, READONLY
PINSEL0     EQU     0xE002C000      ; controls the function of the pins
U0START     EQU     0xE000C000      ; start of UART0 registers
LCR0        EQU     0xC             ; line control register for UART0
LSR0        EQU     0x14            ; line status register for UART0
RAMSTART    EQU     0x40000020      ; start of onboard RAM for 2104
            ENTRY
start
            LDR     sp, = RAMSTART  ; set up stack pointer
            BL      UARTConfig      ; initialize/configure UART0

            ; 1-(3)-(a)
            LDR     r3, =0x40000070
            LDR     r1, = StudentData   ; starting address of characters
            ADD     r1, #43
            MOV     r2, #44
            MOV     r4, #0
Loop
            LDRB    r5, [r1], #-1
            STRB    r5, [r3], #1
            SUB     r2, #1
            CMP     r2, #0
            STRB    r4, [r3]
            BNE     Loop

            ; 1-(3)-(b)
            LDR     r1, =0x40000078
            MOV     r2, #10
Loop1
            LDRB    r0, [r1],#1     ; load character, increment address
            CMP     r2,#0           ; null terminated?
            BLNE    Transmit        ; send character to UART
            SUB     r2, #1
            BNE     Loop1           ; continue if not a '0'

            ; 1-(3)-(c)
            MOV     r9, #20
            LDR     r1, =0x400000A0
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- □ DLAB
- □ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x60
- □ Receiver Data Ready (RDR)
- □ Overrun Error (OE)
- □ Parity Error (PE)
- □ Framing Error (FE)
- □ Break Interrupt (BI)
- ☑ Tx Holding Register Empty (THRE)
- ☑ Transmitter Empty (TEMT)
- □ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- □ RBR IE
- □ THRE IE
- □ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01 □ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- □ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0E
- U0DLM: 0x00
- Baudrate: 13392

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

按F5的結果會全部輸出出來(紅色底線)，因為程式執行到最後，將Transmit Holding Register和Transmit Shift Register的值傳送至UART#1。(有模擬輸出端)

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST1-3\\Objec
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

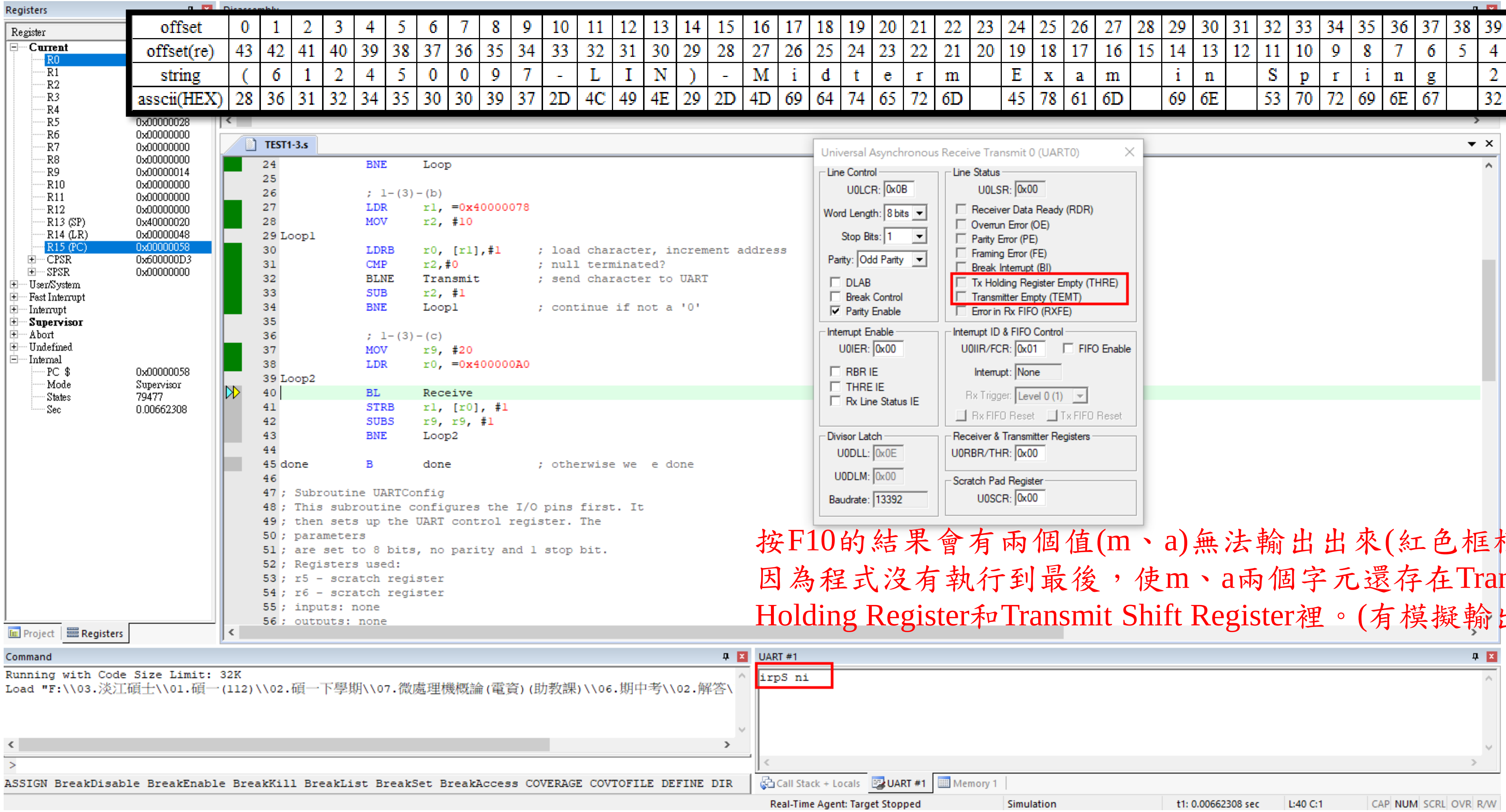
UART #1:

```
irpS ni ma
```

Real-Time Agent: Not in target   Simulation   t1: 64.43034475 sec   L:9 C:1   CAP NUM SCRL OVR R/W

(3) (c) to use calls to subroutine **Receive** to receive a sequence of **error-free 20** characters from the UART0 and put them in memory starting from address 0x400000**A0**.(F10未進Receive)

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

Register	
Current	
R0	
R1	
R2	
R3	
R4	
R5	0x00000028
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000014
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x40000020
R14 (LR)	0x00000048
R15 (PC)	0x00000058
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000058
Mode	Supervisor
States	79477
Sec	0.00662308

TEST1-3.s

```
                BNE         Loop

                ; 1-(3)-(b)
                LDR         r1, =0x40000078
                MOV         r2, #10
Loop1
                LDRB        r0, [r1],#1         ; load character, increment address
                CMP         r2,#0               ; null terminated?
                BLNE        Transmit            ; send character to UART
                SUB         r2, #1
                BNE         Loop1               ; continue if not a '0'

                ; 1-(3)-(c)
                MOV         r9, #20
                LDR         r0, =0x400000A0
Loop2
                BL          Receive
                STRB        r1, [r0], #1
                SUBS        r9, r9, #1
                BNE         Loop2

done            B           done                ; otherwise we  e done

; Subroutine UARTConfig
; This subroutine configures the I/O pins first. It
; then sets up the UART control register. The
; parameters
; are set to 8 bits, no parity and 1 stop bit.
; Registers used:
; r5 - scratch register
; r6 - scratch register
; inputs: none
; outputs: none
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x00
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
□ Tx Holding Register Empty (THRE)
□ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01 □ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

按F10的結果會有兩個值(m、a)無法輸出出來(紅色框框)，因為程式沒有執行到最後，使m、a兩個字元還存在Transmit Holding Register和Transmit Shift Register裡。(有模擬輸出端)

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR
```

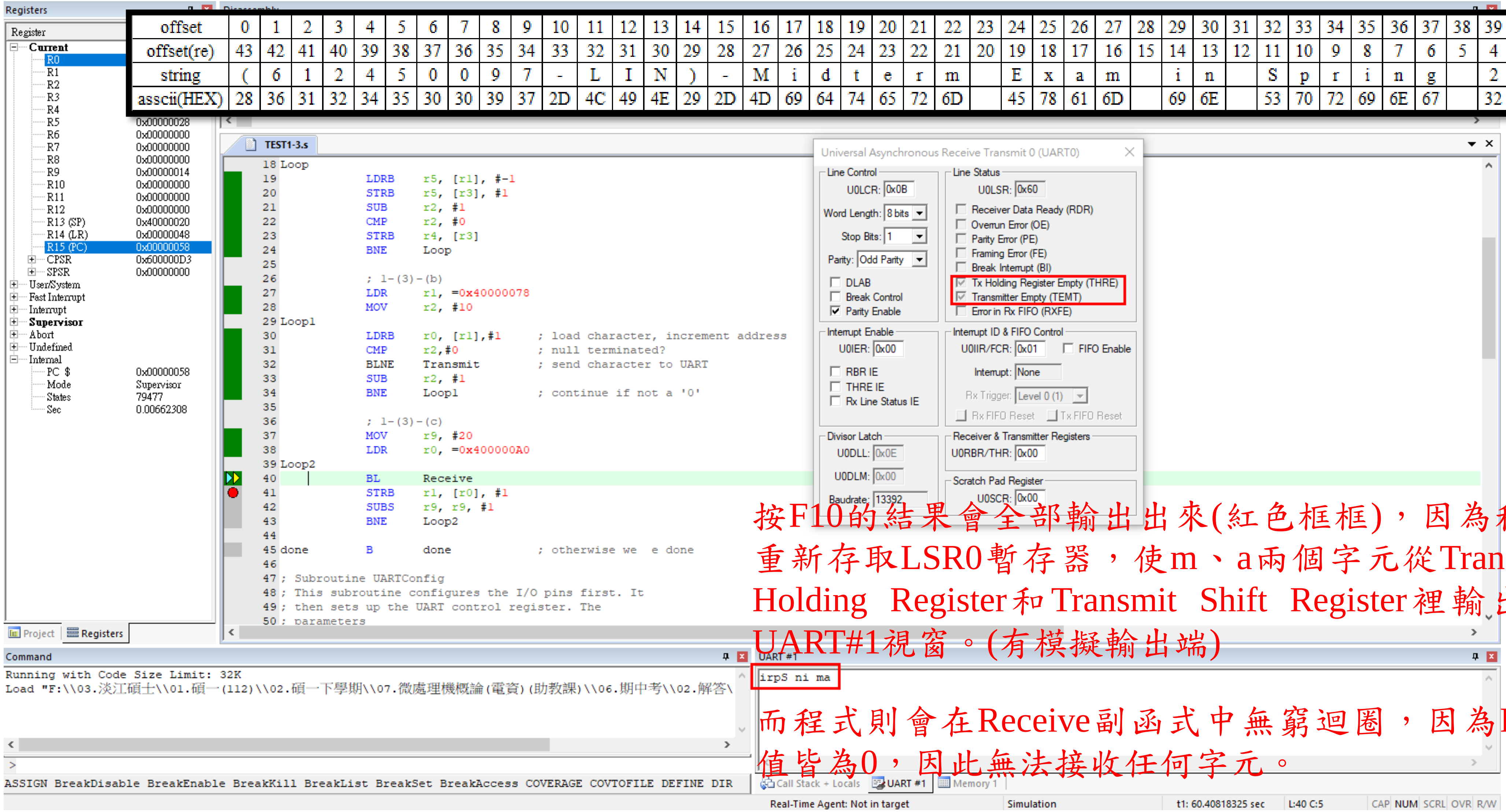
UART #1

```
irpS ni
```

Real-Time Agent: Target Stopped | Simulation | t1: 0.00662308 sec | L:40 C:1 | CAP NUM SCRL OVR R/W

(3) (c) to use calls to subroutine **Receive** to receive a sequence of **error-free 20** characters from the UART0 and put them in memory starting from address 0x400000**A0**.(F10)

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



Registers

Register	
Current	
R0	
R1	
R2	
R3	
R4	
R5	0x00000028
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000014
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x40000020
R14 (LR)	0x00000048
R15 (PC)	0x00000058
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x00000058
Mode	Supervisor
States	79477
Sec	0.00662308

TEST1-3.s

```
Loop
                LDRB    r5, [r1], #-1
                STRB    r5, [r3], #1
                SUB     r2, #1
                CMP     r2, #0
                STRB    r4, [r3]
                BNE     Loop

                ; 1-(3)-(b)
                LDR     r1, =0x40000078
                MOV     r2, #10
Loop1
                LDRB    r0, [r1],#1         ; load character, increment address
                CMP     r2,#0               ; null terminated?
                BLNE    Transmit            ; send character to UART
                SUB     r2, #1
                BNE     Loop1               ; continue if not a '0'

                ; 1-(3)-(c)
                MOV     r9, #20
                LDR     r0, =0x400000A0
Loop2
                BL      Receive
                STRB    r1, [r0], #1
                SUBS    r9, r9, #1
                BNE     Loop2

done            B       done                ; otherwise we  e done

; Subroutine UARTConfig
; This subroutine configures the I/O pins first. It
; then sets up the UART control register. The
; parameters
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- □ DLAB
- □ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x60
- □ Receiver Data Ready (RDR)
- □ Overrun Error (OE)
- □ Parity Error (PE)
- □ Framing Error (FE)
- □ Break Interrupt (BI)
- ☑ Tx Holding Register Empty (THRE)
- ☑ Transmitter Empty (TEMT)
- □ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- □ RBR IE
- □ THRE IE
- □ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01
- □ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- □ Rx FIFO Reset
- □ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0E
- U0DLM: 0x00
- Baudrate: 13392

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR
```

UART #1

```
irpS ni ma
```

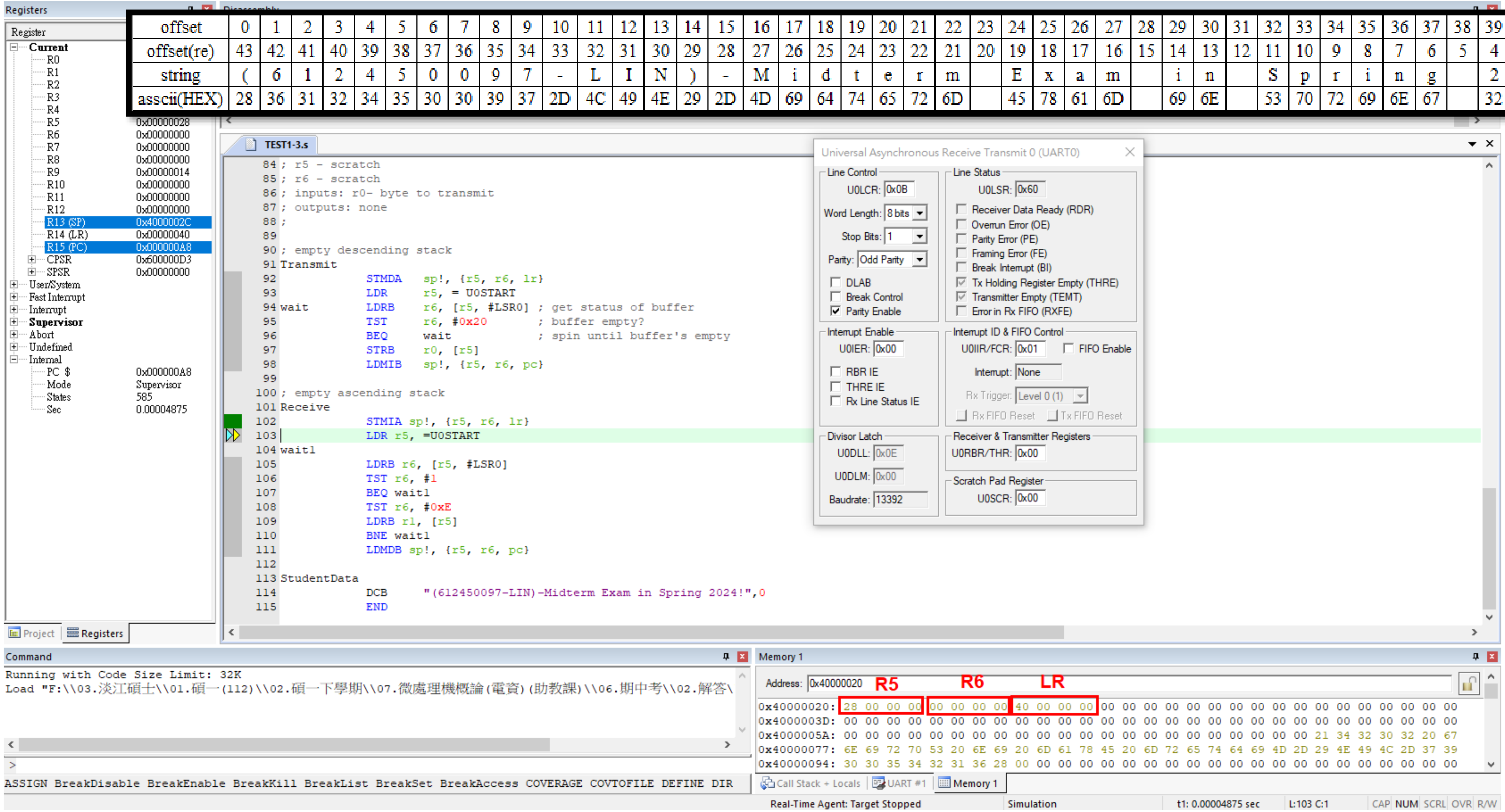
Real-Time Agent: Not in target   Simulation   t1: 60.40818325 sec   L:40 C:5

按F10的結果會全部輸出出來(紅色框框)，因為程式重新存取LSR0暫存器，使m、a兩個字元從Transmit Holding Register和Transmit Shift Register裡輸出到UART#1視窗。(有模擬輸出端)

而程式則會在Receive副函式中無窮迴圈，因為PDR值皆為0，因此無法接收任何字元。

(3) (c) to use calls to subroutine **Receive** to receive a sequence of **error-free 20** characters from the UART0 and put them in memory starting from address 0x400000**A0**.(F11)

(highlight the stack elements with the related registers stored for subroutines Receive and Transmit)



offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21

Registers

Register	Value
Current	
R0	
R1	
R2	
R3	
R4	
R5	0x00000028
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000014
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x4000002C
R14 (LR)	0x00000040
R15 (PC)	0x000000A8
CPSR	0x600000D3
SPSR	0x00000000
User/System	
Fast Interrupt	
Interrupt	
Supervisor	
Abort	
Undefined	
Internal	
PC $	0x000000A8
Mode	Supervisor
States	585
Sec	0.00004875

TEST1-3.s

```
; r5 - scratch
; r6 - scratch
; inputs: r0- byte to transmit
; outputs: none
;

; empty descending stack
Transmit
                STMDA   sp!, {r5, r6, lr}
                LDR     r5, = U0START
wait            LDRB    r6, [r5, #LSR0] ; get status of buffer
                TST     r6, #0x20       ; buffer empty?
                BEQ     wait            ; spin until buffer's empty
                STRB    r0, [r5]
                LDMIB   sp!, {r5, r6, pc}

; empty ascending stack
Receive
                STMIA sp!, {r5, r6, lr}
                LDR r5, =U0START
wait1
                LDRB r6, [r5, #LSR0]
                TST r6, #1
                BEQ wait1
                TST r6, #0xE
                LDRB r1, [r5]
                BNE wait1
                LDMDB sp!, {r5, r6, pc}

StudentData
                DCB     "(612450097-LIN)-Midterm Exam in Spring 2024!",0
                END
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
U0LCR: 0x0B
Word Length: 8 bits
Stop Bits: 1
Parity: Odd Parity
□ DLAB
□ Break Control
☑ Parity Enable

Line Status
U0LSR: 0x60
□ Receiver Data Ready (RDR)
□ Overrun Error (OE)
□ Parity Error (PE)
□ Framing Error (FE)
□ Break Interrupt (BI)
☑ Tx Holding Register Empty (THRE)
☑ Transmitter Empty (TEMT)
□ Error in Rx FIFO (RXFE)

Interrupt Enable
U0IER: 0x00
□ RBR IE
□ THRE IE
□ Rx Line Status IE

Interrupt ID & FIFO Control
U0IIR/FCR: 0x01
□ FIFO Enable
Interrupt: None
Rx Trigger: Level 0 (1)
□ Rx FIFO Reset
□ Tx FIFO Reset

Divisor Latch
U0DLL: 0x0E
U0DLM: 0x00
Baudrate: 13392

Receiver & Transmitter Registers
U0RBR/THR: 0x00

Scratch Pad Register
U0SCR: 0x00

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR
```

Memory 1

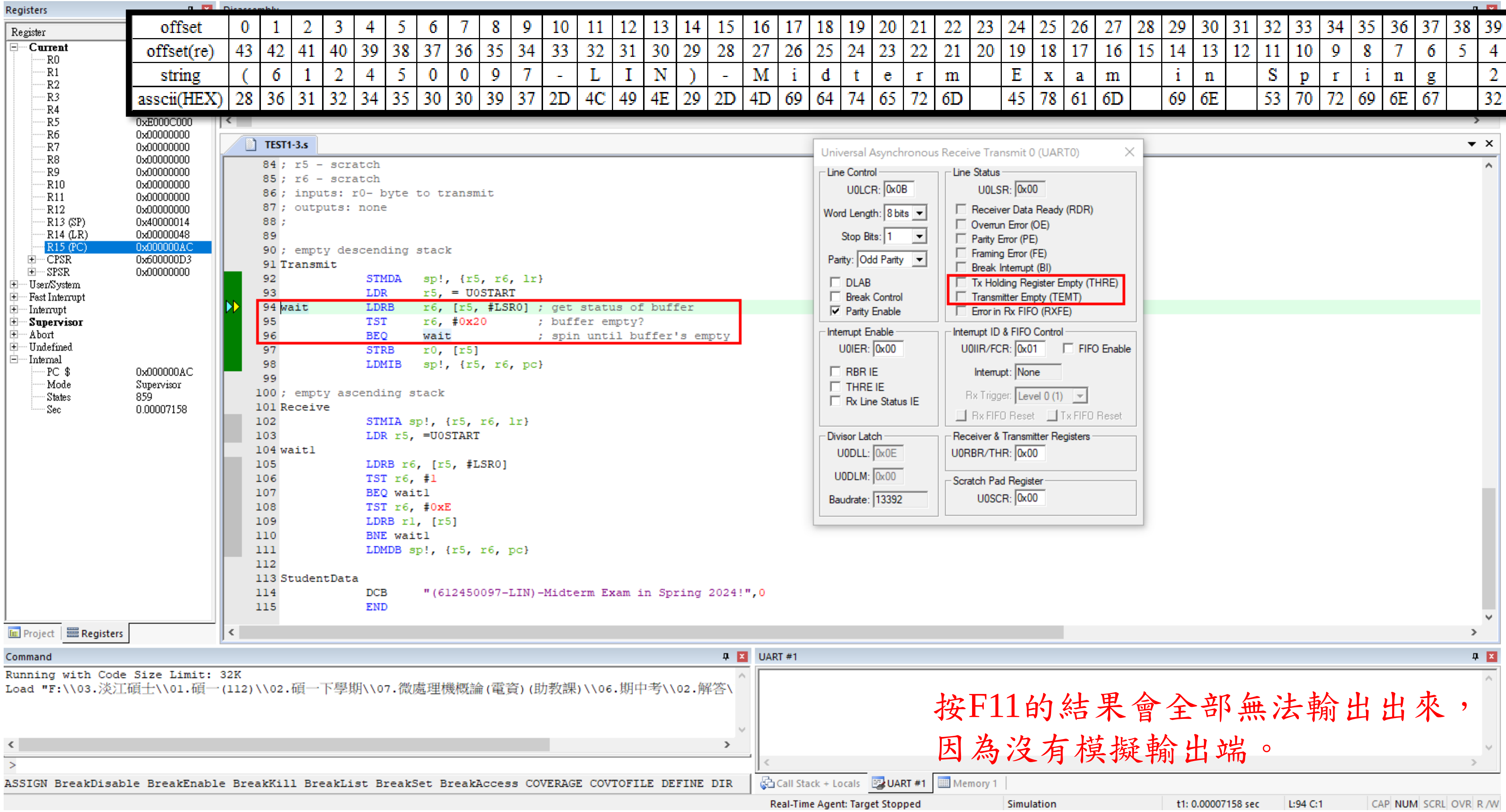
Address: 0x40000020

```
0x40000020: 28 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000003D: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x4000005A: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 21 34 32 30 32 20 67
0x40000077: 6E 69 72 70 53 20 6E 69 20 6D 61 78 45 20 6D 72 65 74 64 69 4D 2D 29 4E 49 4C 2D 37 39
0x40000094: 30 30 35 34 32 31 36 28 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Real-Time Agent: Target Stopped | Simulation | t1: 0.00004875 sec | L:103 C:1 | CAP NUM SCRL OVR R/W

(3) (c) to use calls to subroutine **Receive** to receive a sequence of **error-free 20** characters from the UART0 and put them in memory starting from address 0x400000**A0**.(F11)

offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
offset(re)	43	42	41	40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
string	(	6	1	2	4	5	0	0	9	7	-	L	I	N	)	-	M	i	d	t	e	r	m		E	x	a	m		i	n		S	p	r	i	n	g		2	0	2	4	!
asscii(HEX)	28	36	31	32	34	35	30	30	39	37	2D	4C	49	4E	29	2D	4D	69	64	74	65	72	6D		45	78	61	6D		69	6E		53	70	72	69	6E	67		32	30	32	34	21



```
Register
Current
  R0
  R1
  R2
  R3
  R4
  R5          0xE000C000
  R6          0x00000000
  R7          0x00000000
  R8          0x00000000
  R9          0x00000000
  R10         0x00000000
  R11         0x00000000
  R12         0x00000000
  R13 (SP)    0x40000014
  R14 (LR)    0x00000048
  R15 (PC)    0x000000AC
  CPSR        0x600000D3
  SPSR        0x00000000
User/System
Fast Interrupt
Interrupt
Supervisor
Abort
Undefined
Internal
  PC  $       0x000000AC
  Mode        Supervisor
  States      859
  Sec         0.00007158
```

```
; r5 - scratch
; r6 - scratch
; inputs: r0- byte to transmit
; outputs: none
;

; empty descending stack
Transmit
                STMDA   sp!, {r5, r6, lr}
                LDR     r5, = U0START
wait            LDRB    r6, [r5, #LSR0] ; get status of buffer
                TST     r6, #0x20       ; buffer empty?
                BEQ     wait            ; spin until buffer's empty
                STRB    r0, [r5]
                LDMIB   sp!, {r5, r6, pc}

; empty ascending stack
Receive
                STMIA sp!, {r5, r6, lr}
                LDR r5, =U0START
wait1
                LDRB r6, [r5, #LSR0]
                TST r6, #1
                BEQ wait1
                TST r6, #0xE
                LDRB r1, [r5]
                BNE wait1
                LDMDB sp!, {r5, r6, pc}

StudentData
                DCB     "(612450097-LIN)-Midterm Exam in Spring 2024!",0
                END
```

Universal Asynchronous Receive Transmit 0 (UART0)

Line Control
- U0LCR: 0x0B
- Word Length: 8 bits
- Stop Bits: 1
- Parity: Odd Parity
- □ DLAB
- □ Break Control
- ☑ Parity Enable

Line Status
- U0LSR: 0x00
- □ Receiver Data Ready (RDR)
- □ Overrun Error (OE)
- □ Parity Error (PE)
- □ Framing Error (FE)
- □ Break Interrupt (BI)
- □ Tx Holding Register Empty (THRE)
- □ Transmitter Empty (TEMT)
- □ Error in Rx FIFO (RXFE)

Interrupt Enable
- U0IER: 0x00
- □ RBR IE
- □ THRE IE
- □ Rx Line Status IE

Interrupt ID & FIFO Control
- U0IIR/FCR: 0x01 □ FIFO Enable
- Interrupt: None
- Rx Trigger: Level 0 (1)
- □ Rx FIFO Reset □ Tx FIFO Reset

Divisor Latch
- U0DLL: 0x0E
- U0DLM: 0x00
- Baudrate: 13392

Receiver & Transmitter Registers
- U0RBR/THR: 0x00

Scratch Pad Register
- U0SCR: 0x00

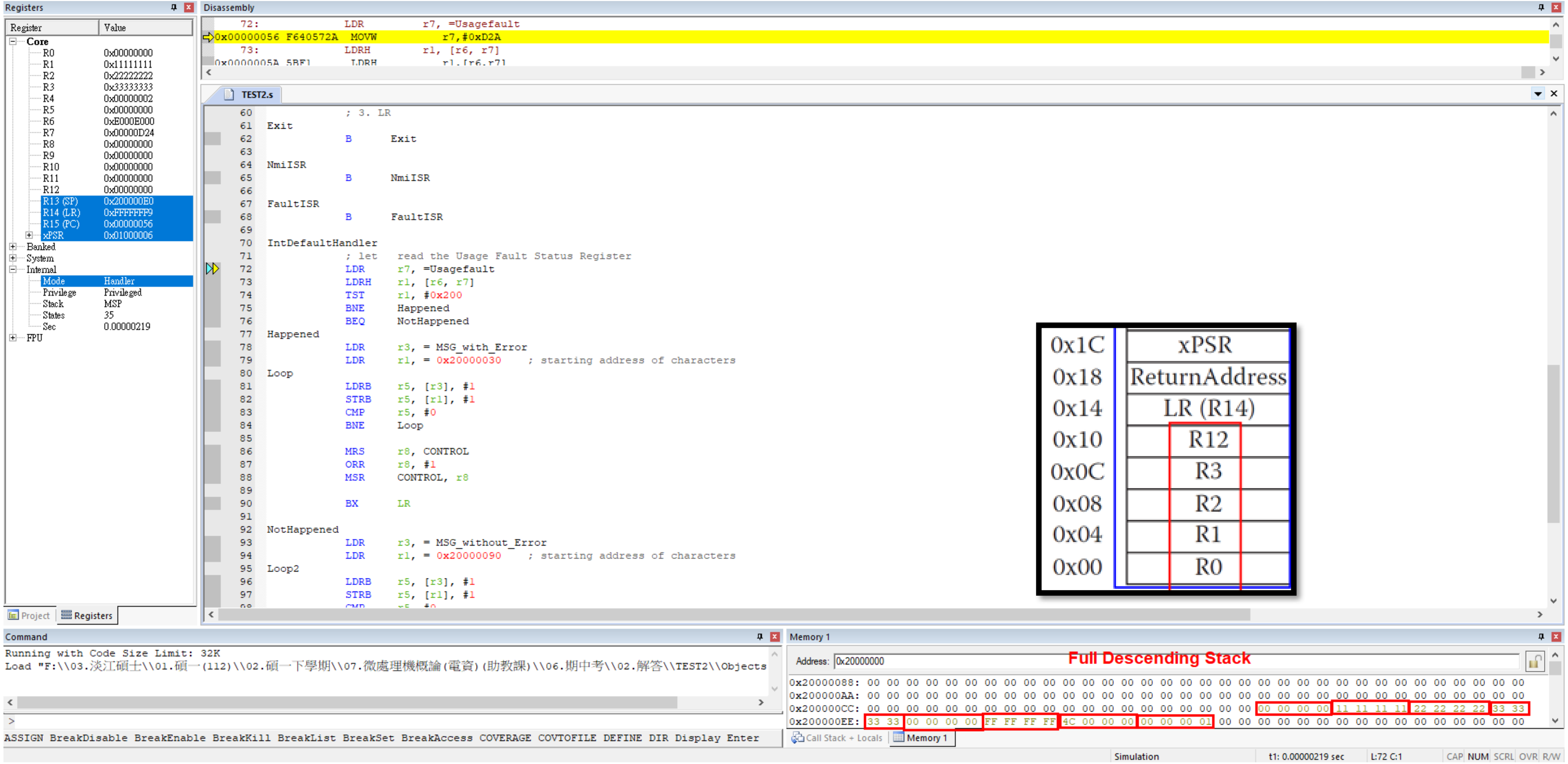
```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\
```

按F11的結果會全部無法輸出出來，因為沒有模擬輸出端。

2. Rewrite Program 15-1 to include the following 2 declarations and

```
MSG_with_Error DCB “DIVIDE-BY-0 Happened!”, 0
MSG_without_Error DCB “DIVIDE-BY-0 Not Happened!”, 0
```

(1) check the usage fault status register, **write string MSG_with_Error to memory with starting address 0x20000030** if a divide-by-zero **has taken place**, and **write string MSG_without_Error to memory with starting address 0x20000090** if a divide-by-zero **has not taken place**.(Be sure to show the **first step (stacking)** in the **entry** sequence upon processor exception and give the **type of the stack** used here.)



Register	Value
Core	
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D24
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000056
xPSR	0x01000006
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	35
Sec	0.00000219
FPU	

```
    72:                LDR           r7, =Usagefault
0x00000056 F640572A  MOVW            r7,#0xD2A
    73:                LDRH          r1, [r6, r7]
0x0000005A 5BF1      LDRH            r1,[r6,r7]
```

```
                ; 3. LR
Exit
                B       Exit

NmiISR
                B       NmiISR

FaultISR
                B       FaultISR

IntDefaultHandler
                ; let    read the Usage Fault Status Register
                LDR      r7, =Usagefault
                LDRH     r1, [r6, r7]
                TST      r1, #0x200
                BNE      Happened
                BEQ      NotHappened
Happened
                LDR      r3, = MSG_with_Error
                LDR      r1, = 0x20000030      ; starting address of characters
Loop
                LDRB     r5, [r3], #1
                STRB     r5, [r1], #1
                CMP      r5, #0
                BNE      Loop

                MRS      r8, CONTROL
                ORR      r8, #1
                MSR      CONTROL, r8

                BX       LR

NotHappened
                LDR      r3, = MSG_without_Error
                LDR      r1, = 0x20000090      ; starting address of characters
Loop2
                LDRB     r5, [r3], #1
                STRB     r5, [r1], #1
                CMP      r5, #0
```

0x1C	xPSR
0x18	ReturnAddress
0x14	LR (R14)
0x10	R12
0x0C	R3
0x08	R2
0x04	R1
0x00	R0

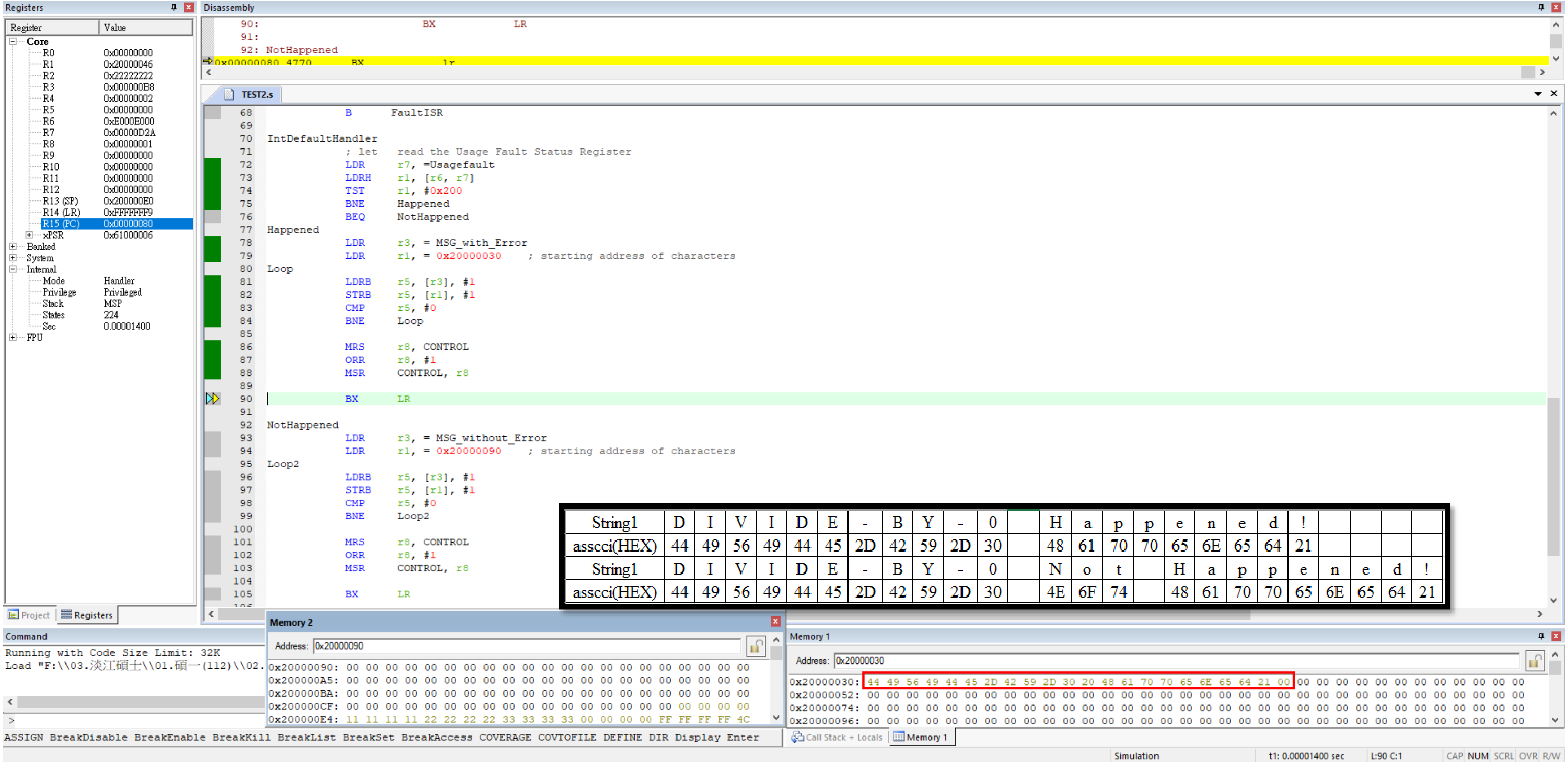
```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.碩一下學期\\07.微處理機概論(電資)(助教課)\\06.期中考\\02.解答\\TEST2\\Objects
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Address: 0x20000000 **Full Descending Stack**

```
0x20000088: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000AA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CC: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 11 11 11 11 22 22 22 22 33 33
0x200000EE: 33 33 00 00 00 00 FF FF FF FF 4C 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Simulation   t1: 0.00000219 sec   L:72 C:1   CAP NUM SCRL OVR R/W

(1) check the usage fault status register, **write string MSG_with_Error to memory with starting address 0x20000030** if a divide-by-zero **has taken place**, and **write string MSG_without_Error to memory with starting address 0x20000090** if a divide-by-zero **has not taken place**.(Be sure to show the **first step (stacking)** in the **entry** sequence upon processor exception and give the **type of the stack** used here.)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x20000046
R2	0x22222222
R3	0x000000B8
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000001
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000080
xPSR	0x61000006
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	224
Sec	0.00001400
FPU	

Disassembly

```
    90:                 BX          LR
    91:
    92: NotHappened
0x00000080 4770      BX        lr
```

TEST2.s

```
                B       FaultISR

IntDefaultHandler
                ; let   read the Usage Fault Status Register
                LDR     r7, =Usagefault
                LDRH    r1, [r6, r7]
                TST     r1, #0x200
                BNE     Happened
                BEQ     NotHappened
Happened
                LDR     r3, = MSG_with_Error
                LDR     r1, = 0x20000030    ; starting address of characters
Loop
                LDRB    r5, [r3], #1
                STRB    r5, [r1], #1
                CMP     r5, #0
                BNE     Loop

                MRS     r8, CONTROL
                ORR     r8, #1
                MSR     CONTROL, r8

                BX      LR

NotHappened
                LDR     r3, = MSG_without_Error
                LDR     r1, = 0x20000090    ; starting address of characters
Loop2
                LDRB    r5, [r3], #1
                STRB    r5, [r1], #1
                CMP     r5, #0
                BNE     Loop2

                MRS     r8, CONTROL
                ORR     r8, #1
                MSR     CONTROL, r8

                BX      LR
```

String1	D	I	V	I	D	E	-	B	Y	-	0		H	a	p	p	e	n	e	d	!				
asscci(HEX)	44	49	56	49	44	45	2D	42	59	2D	30		48	61	70	70	65	6E	65	64	21				
String1	D	I	V	I	D	E	-	B	Y	-	0		N	o	t		H	a	p	p	e	n	e	d	!
asscci(HEX)	44	49	56	49	44	45	2D	42	59	2D	30		4E	6F	74		48	61	70	70	65	6E	65	64	21

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

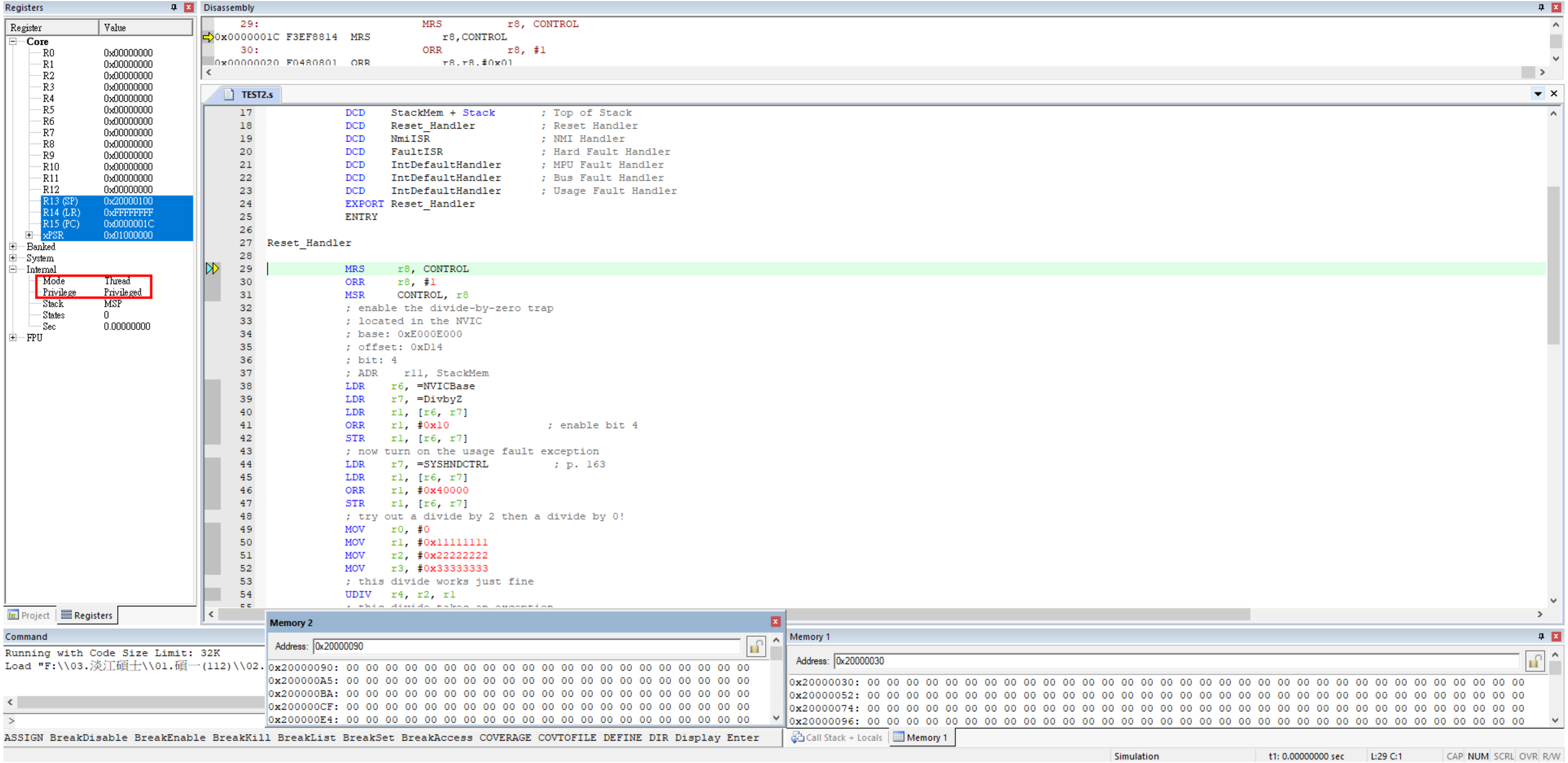
Memory 1

Address: 0x20000030

```
0x20000030: 44 49 56 49 44 45 2D 42 59 2D 30 20 48 61 70 70 65 6E 65 64 21 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(a) from **privileged thread mode** to **unprivileged thread mode**(1/2)



Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000100
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0000001C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	0
Sec	0.00000000
FPU	

```
    29:                         MRS          r8, CONTROL
0x0000001C F3EF8814  MRS           r8,CONTROL
    30:                         ORR          r8, #1
0x00000020 F0480801  ORR           r8,r8,#0x01
```

```
                DCD     StackMem + Stack        ; Top of Stack
                DCD     Reset_Handler           ; Reset Handler
                DCD     NmiISR                  ; NMI Handler
                DCD     FaultISR                ; Hard Fault Handler
                DCD     IntDefaultHandler       ; MPU Fault Handler
                DCD     IntDefaultHandler       ; Bus Fault Handler
                DCD     IntDefaultHandler       ; Usage Fault Handler
                EXPORT Reset_Handler
                ENTRY

Reset_Handler

                MRS     r8, CONTROL
                ORR     r8, #1
                MSR     CONTROL, r8
                ; enable the divide-by-zero trap
                ; located in the NVIC
                ; base: 0xE000E000
                ; offset: 0xD14
                ; bit: 4
                ; ADR    r11, StackMem
                LDR     r6, =NVICBase
                LDR     r7, =DivbyZ
                LDR     r1, [r6, r7]
                ORR     r1, #0x10               ; enable bit 4
                STR     r1, [r6, r7]
                ; now turn on the usage fault exception
                LDR     r7, =SYSHNDCTRL           ; p. 163
                LDR     r1, [r6, r7]
                ORR     r1, #0x40000
                STR     r1, [r6, r7]
                ; try out a divide by 2 then a divide by 0!
                MOV     r0, #0
                MOV     r1, #0x11111111
                MOV     r2, #0x22222222
                MOV     r3, #0x33333333
                ; this divide works just fine
                UDIV    r4, r2, r1
                ; this divide takes an exception
```

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
```

```
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2 — Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

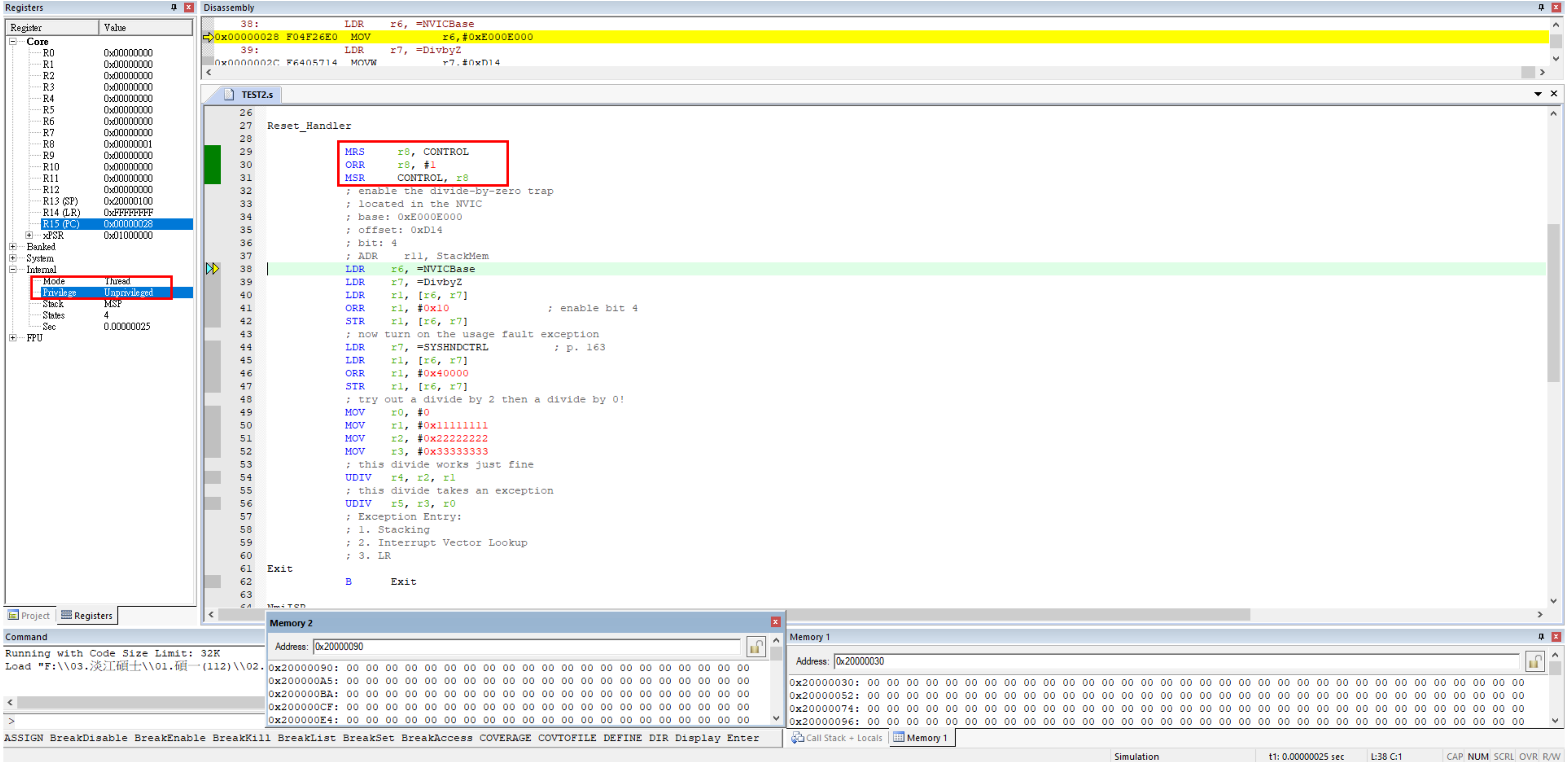
Memory 1 — Address: 0x20000030

```
0x20000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
Simulation    t1: 0.00000000 sec    L:29 C:1    CAP NUM SCRL OVR R/W
```

(2) to switch modes and show the mode changes

(a) from **privileged thread mode** to **unprivileged thread mode**(2/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000001
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000100
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x00000028
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Unprivileged
Stack	MSP
States	4
Sec	0.00000025
FPU	

Disassembly

```
    38:                 LDR     r6, =NVICBase
0x00000028 F04F26E0  MOV          r6,#0xE000E000
    39:                 LDR     r7, =DivbyZ
0x0000002C F6405714  MOVW         r7,#0xD14
```

TEST2.s

```

Reset_Handler

            MRS     r8, CONTROL
            ORR     r8, #1
            MSR     CONTROL, r8
            ; enable the divide-by-zero trap
            ; located in the NVIC
            ; base: 0xE000E000
            ; offset: 0xD14
            ; bit: 4
            ; ADR    r11, StackMem
            LDR     r6, =NVICBase
            LDR     r7, =DivbyZ
            LDR     r1, [r6, r7]
            ORR     r1, #0x10                 ; enable bit 4
            STR     r1, [r6, r7]
            ; now turn on the usage fault exception
            LDR     r7, =SYSHNDCTRL            ; p. 163
            LDR     r1, [r6, r7]
            ORR     r1, #0x40000
            STR     r1, [r6, r7]
            ; try out a divide by 2 then a divide by 0!
            MOV     r0, #0
            MOV     r1, #0x11111111
            MOV     r2, #0x22222222
            MOV     r3, #0x33333333
            ; this divide works just fine
            UDIV    r4, r2, r1
            ; this divide takes an exception
            UDIV    r5, r3, r0
            ; Exception Entry:
            ; 1. Stacking
            ; 2. Interrupt Vector Lookup
            ; 3. LR
Exit
            B       Exit

NmiISR
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

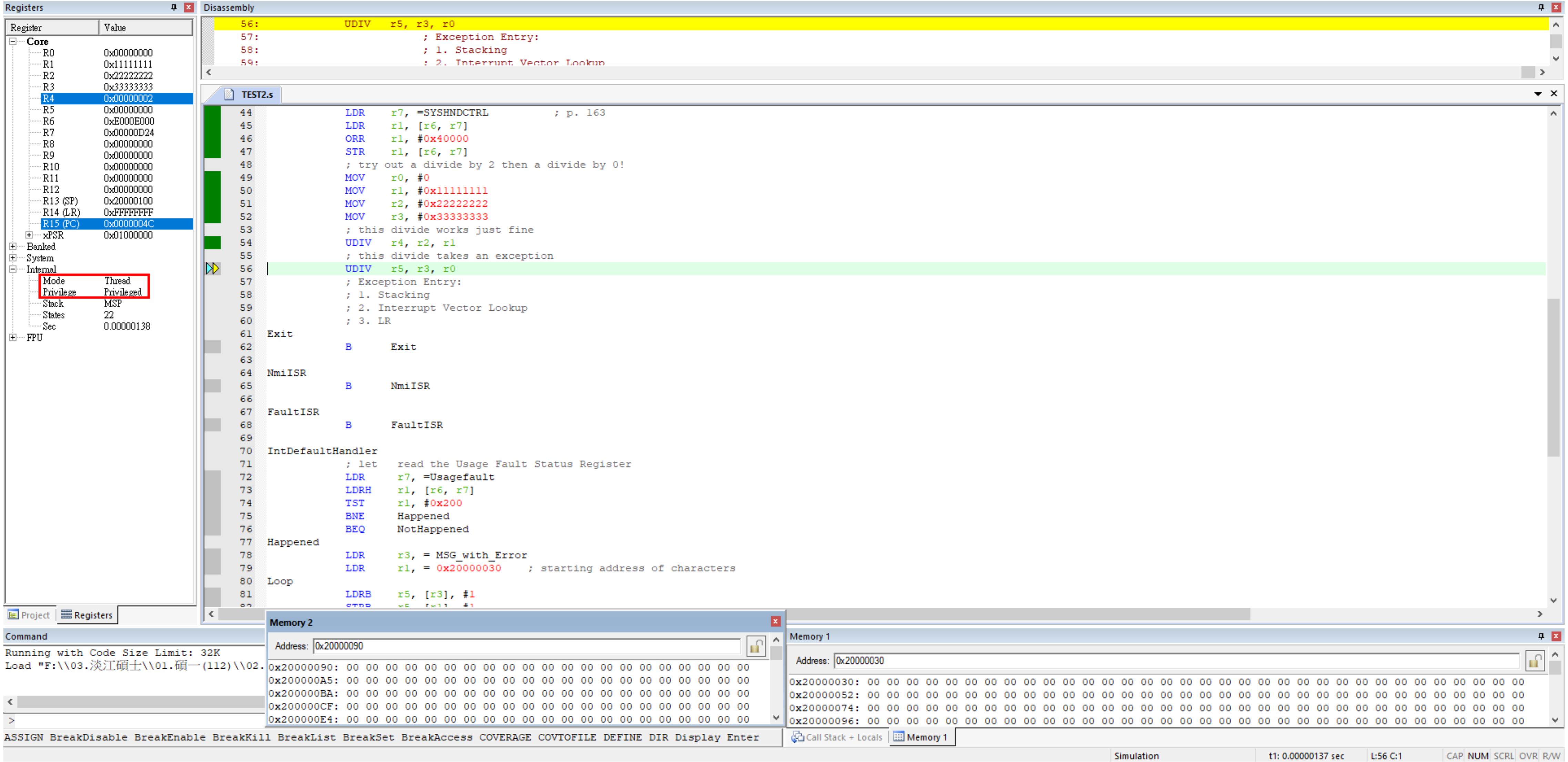
Memory 1

Address: 0x20000030

```
0x20000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(b) from **privileged thread mode** to **privileged handler mode**(1/2)



Registers

```
Register        Value
⊟ Core
    R0          0x00000000
    R1          0x11111111
    R2          0x22222222
    R3          0x33333333
    R4          0x00000002
    R5          0x00000000
    R6          0xE000E000
    R7          0x00000D24
    R8          0x00000000
    R9          0x00000000
    R10         0x00000000
    R11         0x00000000
    R12         0x00000000
    R13 (SP)    0x20000100
    R14 (LR)    0xFFFFFFFF
    R15 (PC)    0x0000004C
  ⊞ xPSR        0x01000000
⊞ Banked
⊞ System
⊟ Internal
    Mode        Thread
    Privilege   Privileged
    Stack       MSP
    States      22
    Sec         0.00000138
⊞ FPU
```

Disassembly

```
    56:                  UDIV    r5, r3, r0
    57:                          ; Exception Entry:
    58:                          ; 1. Stacking
    59:                          ; 2. Interrupt Vector Lookup
```

TEST2.s

```
                LDR     r7, =SYSHNDCTRL         ; p. 163
                LDR     r1, [r6, r7]
                ORR     r1, #0x40000
                STR     r1, [r6, r7]
                ; try out a divide by 2 then a divide by 0!
                MOV     r0, #0
                MOV     r1, #0x11111111
                MOV     r2, #0x22222222
                MOV     r3, #0x33333333
                ; this divide works just fine
                UDIV    r4, r2, r1
                ; this divide takes an exception
                UDIV    r5, r3, r0
                ; Exception Entry:
                ; 1. Stacking
                ; 2. Interrupt Vector Lookup
                ; 3. LR
Exit
                B       Exit

NmiISR
                B       NmiISR

FaultISR
                B       FaultISR

IntDefaultHandler
                ; let   read the Usage Fault Status Register
                LDR     r7, =Usagefault
                LDRH    r1, [r6, r7]
                TST     r1, #0x200
                BNE     Happened
                BEQ     NotHappened
Happened
                LDR     r3, = MSG_with_Error
                LDR     r1, = 0x20000030    ; starting address of characters
Loop
                LDRB    r5, [r3], #1
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2 — Address: 0x20000090

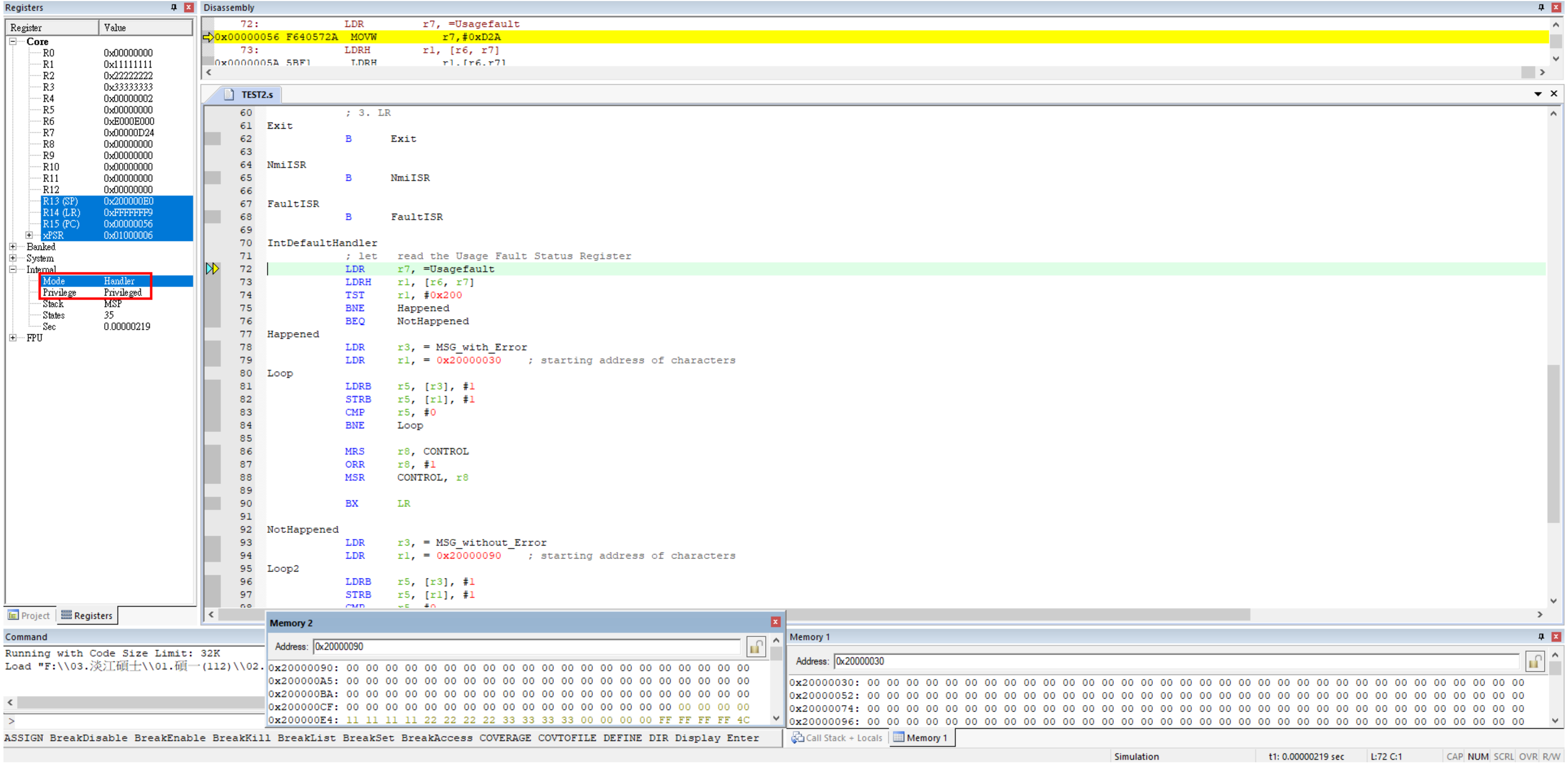
```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Memory 1 — Address: 0x20000030

```
0x20000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(b) from **privileged thread mode** to **privileged handler mode**(2/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D24
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000056
xPSR	0x01000006
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	35
Sec	0.00000219
FPU	

Disassembly

```
    72:               LDR         r7, =Usagefault
0x00000056 F640572A  MOVW            r7,#0xD2A
    73:               LDRH        r1, [r6, r7]
0x0000005A 5BE1       LDRH            r1,[r6,r7]
```

TEST2.s

```
                ; 3. LR
Exit
                B       Exit

NmiISR
                B       NmiISR

FaultISR
                B       FaultISR

IntDefaultHandler
                ; let   read the Usage Fault Status Register
                LDR     r7, =Usagefault
                LDRH    r1, [r6, r7]
                TST     r1, #0x200
                BNE     Happened
                BEQ     NotHappened
Happened
                LDR     r3, = MSG_with_Error
                LDR     r1, = 0x20000030    ; starting address of characters
Loop
                LDRB    r5, [r3], #1
                STRB    r5, [r1], #1
                CMP     r5, #0
                BNE     Loop

                MRS     r8, CONTROL
                ORR     r8, #1
                MSR     CONTROL, r8

                BX      LR

NotHappened
                LDR     r3, = MSG_without_Error
                LDR     r1, = 0x20000090    ; starting address of characters
Loop2
                LDRB    r5, [r3], #1
                STRB    r5, [r1], #1
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

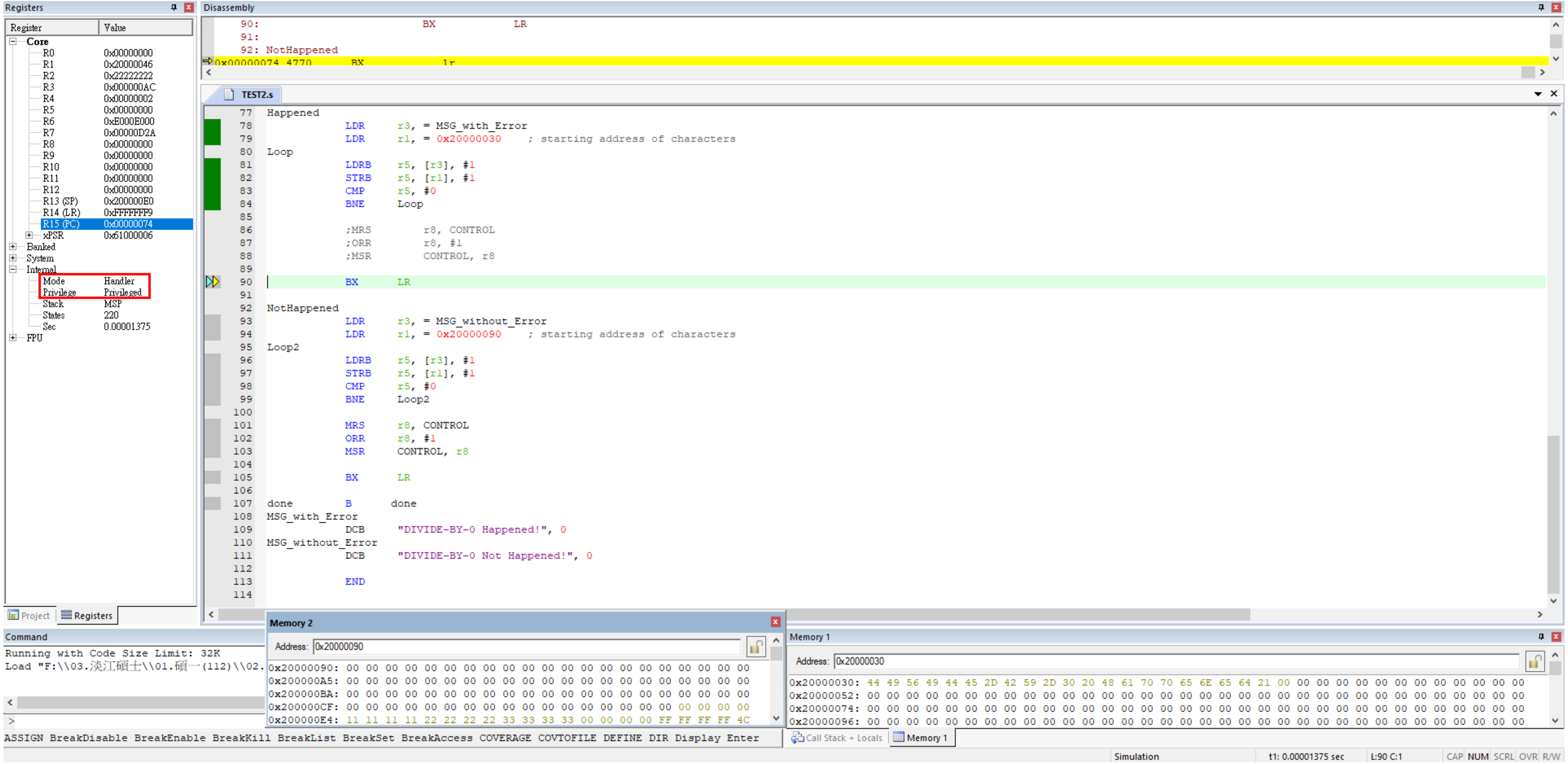
Memory 1

Address: 0x20000030

```
0x20000030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(c) from **privileged handler mode** to **privileged thread mode**(1/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x20000046
R2	0x22222222
R3	0x000000AC
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000074
xPSR	0x61000006
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	220
Sec	0.00001375
FPU	

Disassembly

```
    90:                    BX          LR
    91:
    92: NotHappened
0x00000074 4770      BX          lr
```

TEST2.s

```
Happened
              LDR      r3, = MSG_with_Error
              LDR      r1, = 0x20000030    ; starting address of characters
Loop
              LDRB     r5, [r3], #1
              STRB     r5, [r1], #1
              CMP      r5, #0
              BNE      Loop

              ;MRS         r8, CONTROL
              ;ORR         r8, #1
              ;MSR         CONTROL, r8

              BX       LR

NotHappened
              LDR      r3, = MSG_without_Error
              LDR      r1, = 0x20000090    ; starting address of characters
Loop2
              LDRB     r5, [r3], #1
              STRB     r5, [r1], #1
              CMP      r5, #0
              BNE      Loop2

              MRS      r8, CONTROL
              ORR      r8, #1
              MSR      CONTROL, r8

              BX       LR

done          B      done
MSG_with_Error
              DCB      "DIVIDE-BY-0 Happened!", 0
MSG_without_Error
              DCB      "DIVIDE-BY-0 Not Happened!", 0

              END

```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

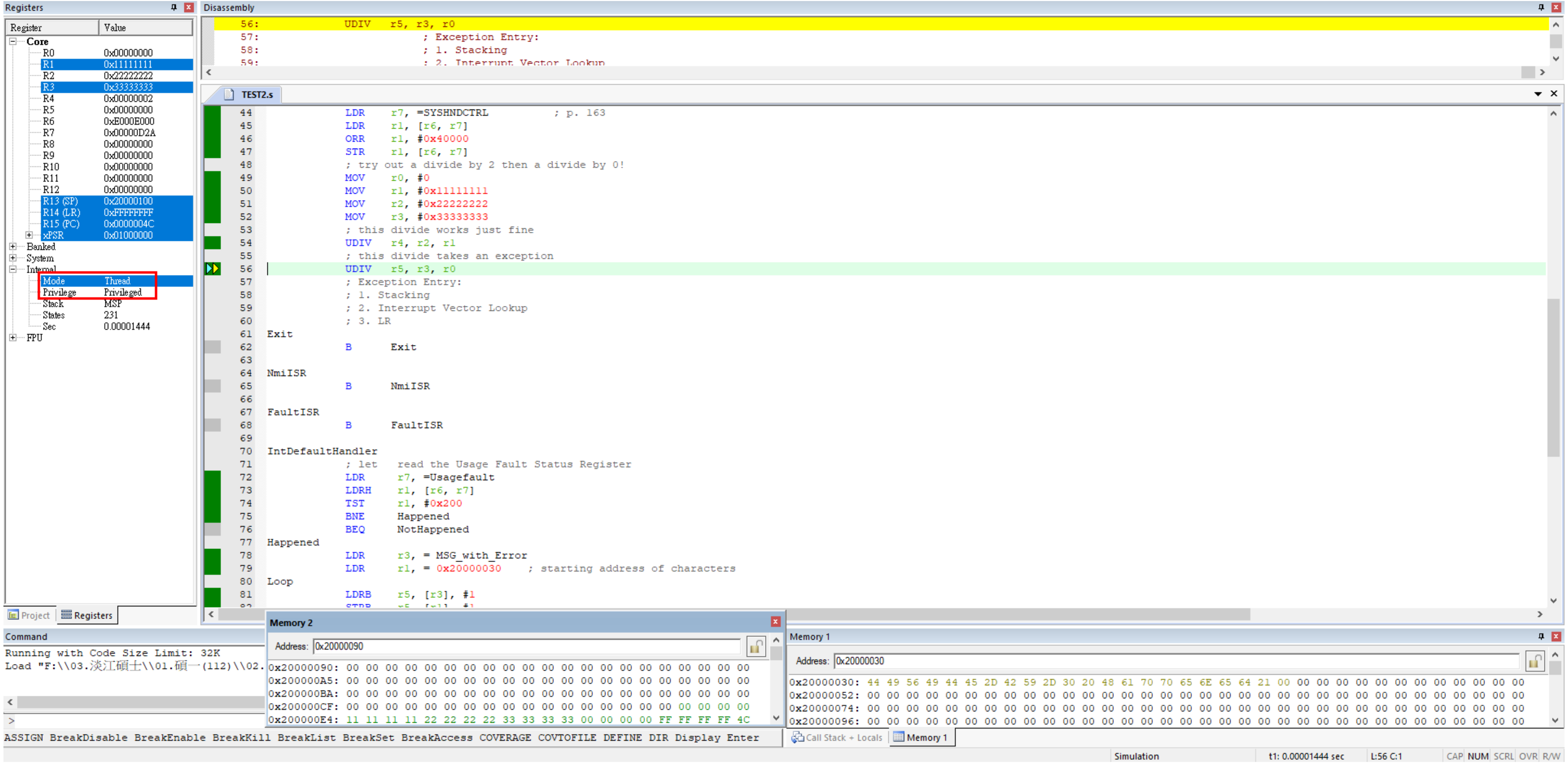
Memory 1

Address: 0x20000030

```
0x20000030: 44 49 56 49 44 45 2D 42 59 2D 30 20 48 61 70 70 65 6E 65 64 21 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(c) from **privileged handler mode** to **privileged thread mode**(2/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000100
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0000004C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Privileged
Stack	MSP
States	231
Sec	0.00001444
FPU	

Disassembly

```
    56:                UDIV     r5, r3, r0
    57:                              ; Exception Entry:
    58:                              ; 1. Stacking
    59:                              ; 2. Interrupt Vector Lookup
```

TEST2.s

```
                LDR     r7, =SYSHNDCTRL          ; p. 163
                LDR     r1, [r6, r7]
                ORR     r1, #0x40000
                STR     r1, [r6, r7]
                ; try out a divide by 2 then a divide by 0!
                MOV     r0, #0
                MOV     r1, #0x11111111
                MOV     r2, #0x22222222
                MOV     r3, #0x33333333
                ; this divide works just fine
                UDIV    r4, r2, r1
                ; this divide takes an exception
                UDIV    r5, r3, r0
                ; Exception Entry:
                ; 1. Stacking
                ; 2. Interrupt Vector Lookup
                ; 3. LR
Exit
                B       Exit

NmiISR
                B       NmiISR

FaultISR
                B       FaultISR

IntDefaultHandler
                ; let   read the Usage Fault Status Register
                LDR     r7, =Usagefault
                LDRH    r1, [r6, r7]
                TST     r1, #0x200
                BNE     Happened
                BEQ     NotHappened
Happened
                LDR     r3, = MSG_with_Error
                LDR     r1, = 0x20000030     ; starting address of characters
Loop
                LDRB    r5, [r3], #1
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2 — Address: 0x20000090

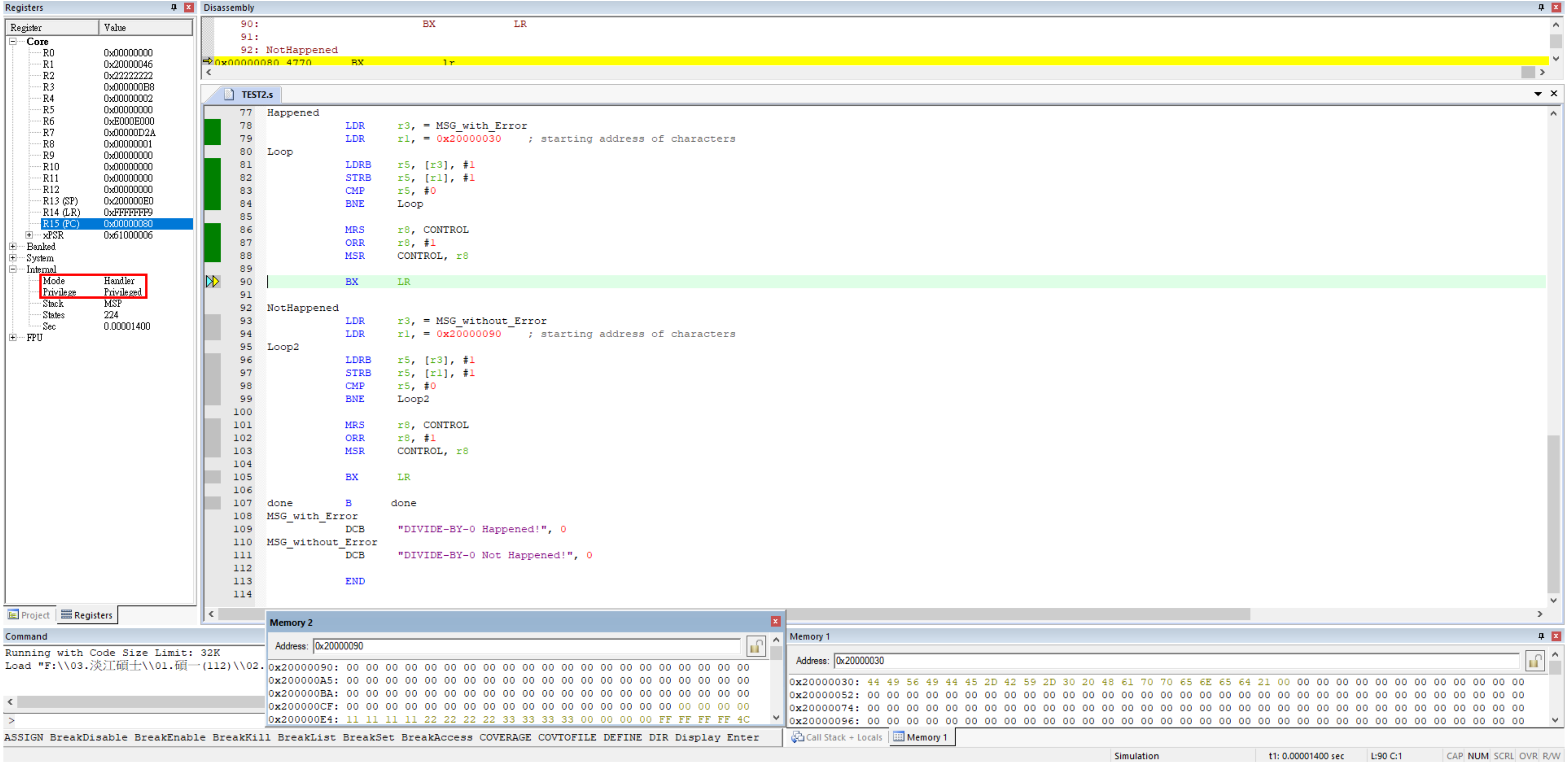
```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

Memory 1 — Address: 0x20000030

```
0x20000030: 44 49 56 49 44 45 2D 42 59 2D 30 20 48 61 70 70 65 6E 65 64 21 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(d) from **privileged handler mode** to **unprivileged thread mode**(1/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x20000046
R2	0x22222222
R3	0x000000B8
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000001
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x200000E0
R14 (LR)	0xFFFFFFF9
R15 (PC)	0x00000080
xPSR	0x61000006
Banked	
System	
Internal	
Mode	Handler
Privilege	Privileged
Stack	MSP
States	224
Sec	0.00001400
FPU	

Disassembly

```
    90:                             BX          LR
    91:
    92: NotHappened
0x00000080 4770      BX          lr
```

TEST2.s

```
Happened
            LDR     r3, = MSG_with_Error
            LDR     r1, = 0x20000030    ; starting address of characters
Loop
            LDRB    r5, [r3], #1
            STRB    r5, [r1], #1
            CMP     r5, #0
            BNE     Loop

            MRS     r8, CONTROL
            ORR     r8, #1
            MSR     CONTROL, r8

            BX      LR

NotHappened
            LDR     r3, = MSG_without_Error
            LDR     r1, = 0x20000090    ; starting address of characters
Loop2
            LDRB    r5, [r3], #1
            STRB    r5, [r1], #1
            CMP     r5, #0
            BNE     Loop2

            MRS     r8, CONTROL
            ORR     r8, #1
            MSR     CONTROL, r8

            BX      LR

done        B       done
MSG_with_Error
            DCB     "DIVIDE-BY-0 Happened!", 0
MSG_without_Error
            DCB     "DIVIDE-BY-0 Not Happened!", 0

            END
```

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

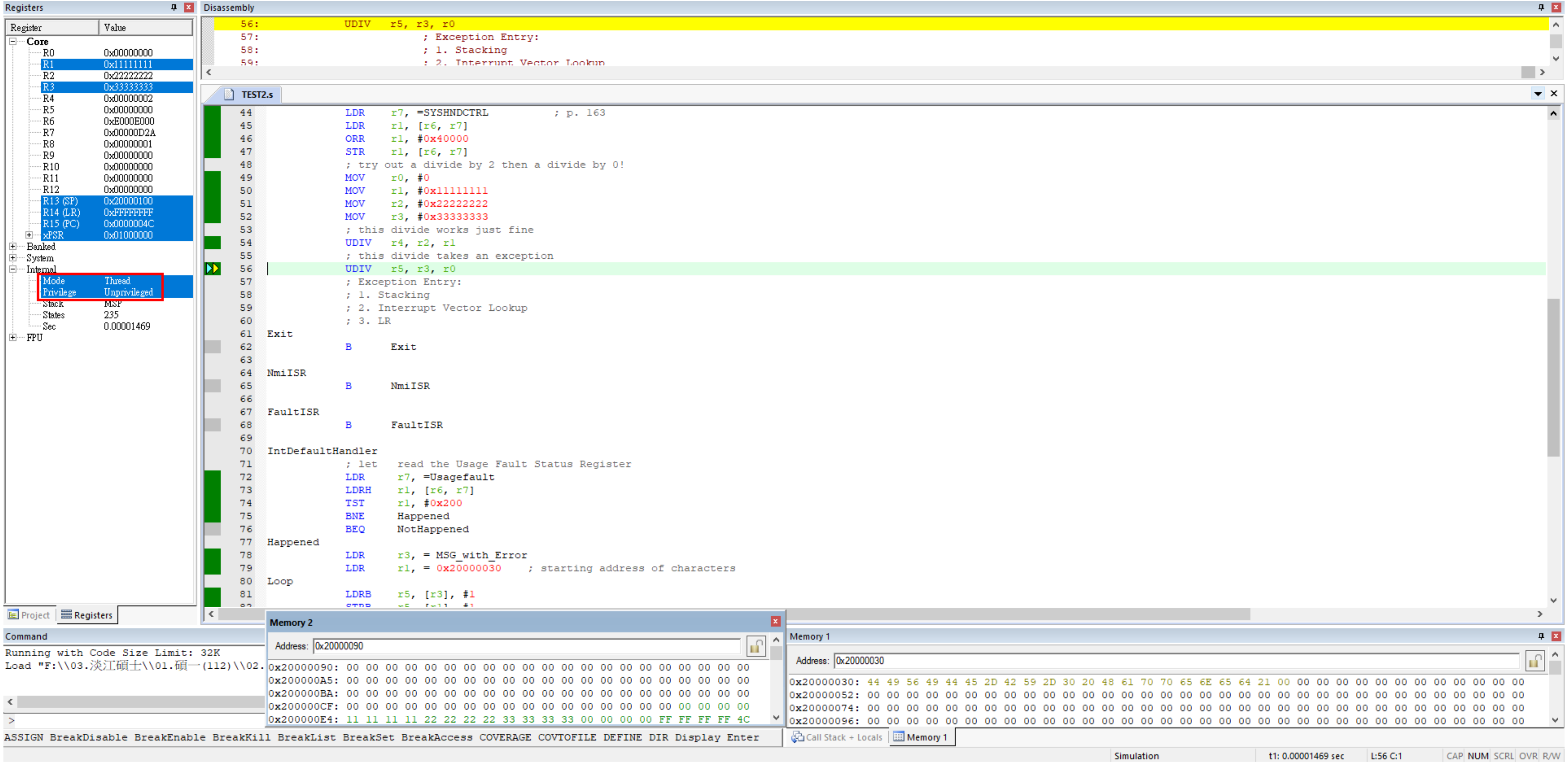
Memory 1

Address: 0x20000030

```
0x20000030: 44 49 56 49 44 45 2D 42 59 2D 30 20 48 61 70 70 65 6E 65 64 21 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

(2) to switch modes and show the mode changes

(d) from **privileged handler mode** to **unprivileged thread mode**(2/2)



Registers

Register	Value
Core	
R0	0x00000000
R1	0x11111111
R2	0x22222222
R3	0x33333333
R4	0x00000002
R5	0x00000000
R6	0xE000E000
R7	0x00000D2A
R8	0x00000001
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x20000100
R14 (LR)	0xFFFFFFFF
R15 (PC)	0x0000004C
xPSR	0x01000000
Banked	
System	
Internal	
Mode	Thread
Privilege	Unprivileged
Stack	MSP
States	235
Sec	0.00001469
FPU	

Disassembly

```
    56:              UDIV     r5, r3, r0
    57:                          ; Exception Entry:
    58:                          ; 1. Stacking
    59:                          ; 2. Interrupt Vector Lookup
```

TEST2.s

```
            LDR     r7, =SYSHNDCTRL          ; p. 163
            LDR     r1, [r6, r7]
            ORR     r1, #0x40000
            STR     r1, [r6, r7]
            ; try out a divide by 2 then a divide by 0!
            MOV     r0, #0
            MOV     r1, #0x11111111
            MOV     r2, #0x22222222
            MOV     r3, #0x33333333
            ; this divide works just fine
            UDIV    r4, r2, r1
            ; this divide takes an exception
            UDIV    r5, r3, r0
            ; Exception Entry:
            ; 1. Stacking
            ; 2. Interrupt Vector Lookup
            ; 3. LR
Exit
            B       Exit

NmiISR
            B       NmiISR

FaultISR
            B       FaultISR

IntDefaultHandler
            ; let    read the Usage Fault Status Register
            LDR      r7, =Usagefault
            LDRH     r1, [r6, r7]
            TST      r1, #0x200
            BNE      Happened
            BEQ      NotHappened
Happened
            LDR      r3, = MSG_with_Error
            LDR      r1, = 0x20000030      ; starting address of characters
Loop
            LDRB     r5, [r3], #1
            STRB     r5, [r1], #1
```

Project | Registers

Command

```
Running with Code Size Limit: 32K
Load "F:\\03.淡江碩士\\01.碩一(112)\\02.
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE COVTOFILE DEFINE DIR Display Enter
```

Memory 2

Address: 0x20000090

```
0x20000090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000A5: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000BA: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000CF: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x200000E4: 11 11 11 11 22 22 22 22 33 33 33 33 00 00 00 00 FF FF FF FF 4C
```

Memory 1

Address: 0x20000030

```
0x20000030: 44 49 56 49 44 45 2D 42 59 2D 30 20 48 61 70 70 65 6E 65 64 21 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000052: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000074: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20000096: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Call Stack + Locals | Memory 1

Simulation | t1: 0.00001469 sec | L:56 C:1 | CAP NUM SCRL OVR R/W

Q&A

Thanks for your attention !!