

Setting/Unsetting Privilege

(not in text book)

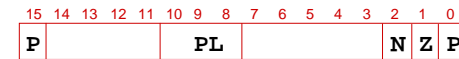
CIT 593

1/8

Privilege

Privilege: Processor modes

- Encoded in 15th bit of processor status register (PSR)
- Privileged (supervisor)
 - Bit 15 = 1
- Unprivileged (user)
 - Bit 15 = 0

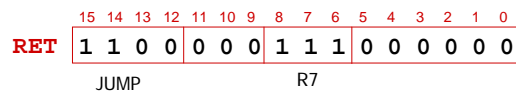


CIT 593

2/8

RET vs. RTT

RET



- Return from subroutine
- Stores PC + 1 in R7, however does not unset the privilege bit
- i.e. PSR[15] remains 1 (Supervisor mode stays on)
- This allows user to access memory locations that it should have access to.

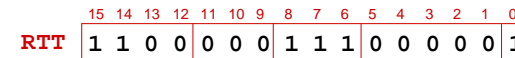
CIT 593

3/8

RET vs. RTT (contd..)

RTT

- Similar to RET
- Except that it clears PSR[15]
- That is brings the system into user mode.
- Use especially for TRAP returns



CIT 593

4/8

TRAP routine implementation flaw in lc3os.asm

lc3os.asm uses RET for TRAP routine return

- This allows user to illegal access to memory portions after return from TRAP

So why not just change RET to RTT

- GETC and OUT work fine
- IN & PUTS trap routine do not work correctly as it calls the other trap routines i.e. GETC and OUT
 - > See the implementation on next slide

CIT 593

5/8

IN Routine

```
TRAP_IN: ST R7,OS_SAVE_R7 ; save R7
GETC      ; read a character
OUT      ; echo back to monitor
LD R7,OS_SAVE_R7 ; restore R7
RTT
```

```
TRAP_GETC:LDI R0,OS_KBSR ; wait for a keystroke
BRzp TRAP_GETC
LDI R0,OS_KBDR ; read char from keyboard & put it in R0
RTT
```

Why will this not work?

- Because returning from GETC will make the machine in unprivileged mode
 - > RTT unsets privilege
- Not good to call a TRAP routine inside another

CIT 593

6/8

IN Rewritten

```
TRAP_IN: ST R7, OS_SAVE_R7
ST R1, OS_SAVE_R1
ST R2, OS_SAVE_R2
;GETC
TRAP_GETC_WAIT: LDI R1,OS_KBSR ; wait for a keystroke
BRzp TRAP_GETC_WAIT
LDI R0,OS_KBDR ;read it and put in R0
;OUT (echo back to monitor)
TRAP_OUT_LOOP: LDI R2,OS_DSR ; wait for display to be ready
BRzp TRAP_OUT_LOOP
STI R0,OS_DDR
LD R1,OS_SAVE_R1 ; restore R1
LD R2,OS_SAVE_R2 ; restore R2
LD R7,OS_SAVE_R7 ; restore R7
RTT
```

CIT 593

7/8

PUTS rewritten

```
TRAP_PUTS: ST R0,OS_SAVE_R0
ST R1,OS_SAVE_R1
ST R7,OS_SAVE_R7
ADD R1,R0,#0
TRAP_PUTS_LOOP: LDR R0,R1,#0
BRz TRAP_PUTS_DONE
OUT
ADD R1,R1,#1
BRnzp TRAP_PUTS_LOOP
TRAP_PUTS_DONE: LD R0,OS_SAVE_R0
LD R1,OS_SAVE_R1
LD R7,OS_SAVE_R7
RET
```

```
TRAP_PUTS: ST R0,OS_SAVE_R0
ST R1,OS_SAVE_R1
ST R7,OS_SAVE_R7
ST R3,OS_SAVE_R3
TRAP_PUTS_LOOP: LDR R1, R0, #0
BRz TRAP_PUTS_DONE
TRAP_PUTS_WAIT: LDI R3, OS_DSR
BRzp TRAP_PUTS_LOOP
STI R1, OS_DDR
ADD R0, R0, #1
BRnzp TRAP_PUTS_LOOP
TRAP_PUTS_DONE: LD R0,OS_SAVE_R0
LD R1,OS_SAVE_R1
LD R7,OS_SAVE_R7
LD R3,OS_SAVE_R3
RTT
```

CIT 593

8/8