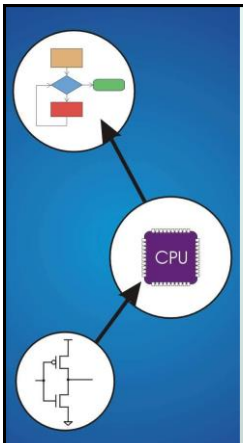


Second Midterm
Review Slides

Original slides from Gregory Byrd, North Carolina State University

Modified slides by Chris Wilcox, Colorado State University

Review Topics

- LC-3 Programming
- Memory Model/Stack Execution

LC-3 Architecture

Instruction Set (First Half)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD+	0001				DR			SR1			0	0	0	SR2		
ADD+	0001				DR			SR1			1	imm5				
AND+	0101				DR			SR1			0	0	0	SR2		
AND+	0101				DR			SR1			1	imm5				
BR	0000	n	z	p	PCoffset9											
JMP	1100				000			BaseR			000000					
JSR	0100	1	PCoffset11													
JSRR	0100	0	0	0	BaseR			000000								
LD+	0010				DR			PCoffset9								

LC-3 Architecture

Instruction Set (Second Half)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LDI+	1010				DR			I	PCoffset9							
LDR+	0110				DR			BaseR			offset6					
LEA+	1110				DR			PCoffset9								
NOT+	1001				DR			SR			111111					
RET	1100				000			111			000000					
RTI	1000	000000000000														
ST	0011				SR			PCoffset9								
STI	1011				SR			PCoffset9								
STR	0111				SR			BaseR			offset6					
TRAP	1111				0000			trapvect8								

LC-3 Architecture
Addressing Modes

- Load -- read data **from memory to register**
  - **LD:** PC-relative mode
  - **LDR:** base+offset mode
  - **LDI:** indirect mode
- Store -- write data **from register to memory**
  - **ST:** PC-relative mode
  - **STR:** base+offset mode
  - **STI:** indirect mode
- Load pointer: **compute address, save in register**
  - **LEA:** immediate mode
  - *does not access memory*

LC-3 Architecture
Machine Code to Assembly

- What is the assembly code for machine instruction `0101010010111101`?
- Step 1) Identify opcode: `0101` = AND
- Step 2) Parse entire instruction (use reference)
- Step 3) Get values from each field

OPCODE	DR	SR	1	imm5
15:12	11:9	8:6	5	4:0
0101	010	010	1	11101
AND	R2	R2		-3

- Step 4) Translate to mnemonics: `AND R2,R2,#-3`

LC-3 Architecture
Assembly to Machine Code

- What is the machine code for assembly instruction `NOT R7,R6`?
- Step 1) Identify opcode: NOT = `1001`
- Step 2) Put values into each field:

NOT	R7	R6	
OPCODE	DR	SR	111111
15:12	11:9	8:6	5:0
1001	111	110	111111

- Step 3) Build machine instruction: `1001111110111111`

LC-3 Architecture
Assembly Code Syntax

```
          .ORIG   x3000
MAIN      AND     R0,R0,#0      ; Initialize Sum
          JSR COMPUTE           ; Call function
          ST      R0, SUM       ; Store Sum
          HALT                  ; Program complete
COMPUTE   LD      R1,OPERAND1   ; Load Operand1
          LD      R2,OPERAND2   ; Load Operand2
          ADD     R0,R1,R2      ; Compute Sum
          RET                   ; Function return
;; Input data set
OPERAND1  .FILL     x1234       ; Operand1
OPERAND2  .FILL     x4321       ; Operand2
SUM       .BLKW     1           ; Sum
          .END
```

Memory Model
Push and Pop Stack

- Assume POP and PUSH code as follows:

```
MACRO PUSH(reg)
        ADD R6,R6,#-1   ; Decrement SP
        STR reg,R6,#0   ; Store value
END

MACRO POP(reg)
        LDR reg,R6,#0   ; Load value
        ADD R6,R6,#1    ; Increment SP
END
```


Memory Model
Detailed Example

- Main program to illustrate stack convention:

```
        .ORIG x3000
MAIN    LD R6,STACK     ; init stack pointer
        LD R1,OPERAND1  ; load second operand
        PUSH R1         ; PUSH second operand
        LD R0,OPERAND0  ; load first operand
        PUSH R0         ; PUSH first operand
        JSR FUNCTION    ; call function
        LDR R0,R6,#0    ; POP return value
        ADD R6,R6,#3    ; unwind stack
        ST  R0,RESULT   ; store result
        HALT
```

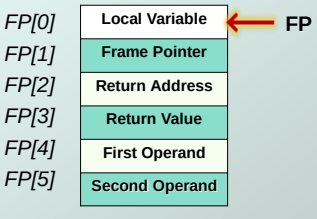

Memory Model
Detailed Example

- Function code to illustrate stack convention:

```
FUNCTION
        ADD R6,R6,#-1   ; alloc return value
        PUSH R7         ; PUSH return address
        PUSH R5         ; PUSH frame pointer
        ADD R5,R6,#-1   ; FP = SP-1

        ADD R6,R6,#-1   ; alloc local variable
        LDR R2,R5,#4    ; load first operand
        LDR R3,R5,#5    ; load second operand
        ADD R4,R3,R2    ; add operands
        STR R4,R5,#0    ; store local variable
```


Memory Model
Detailed Example



Stack before STR instruction

Memory Model
Detailed Example

- Function code to illustrate stack convention:

```
FUNCTION ; stack exit code
        STR R4,R5,#3    ; store return value
        ADD R6,R5,#1    ; SP = FP+1
        POP R5          ; POP frame pointer
        POP R7          ; POP return address
        RET             ; return

OPERAND0   .FILL x1234  ; first operand
OPERAND1   .FILL x2345  ; second operand
RESULT     .BLKW 1      ; result
STACK      .FILL x4000  ; stack address
```