ECE 220 Computer Systems & Programming

Lecture 19 – C to LC-3 with Linked List Data Structure

ECE ILLINOIS

ILLINOIS

Recursive linked list traversal

Problem statement: Convert the following function from C to LC-3. This function recursively traverses a linked list and prints its content.

```
int print_list(node *head)
    {
      if (!head) return 0;
      printf("%c", head->data);
      return print_list(head->next);
    }
```

```
typedef struct tag
{
char data;
struct tag *next;
} node; */
```

Stack Build-up and Tear-down

Caller function	1. Caller setup (push callee's arguments onto stack) 2. Pass control to callee (JSR/JSRR)
Callee function	3. Callee setup (push bookkeeping info and local variables onto stack) 4. Execute function 5. Callee tear-down (update return value, pop local variables, caller's frame pointer and return address from stack) 6. Return to caller (RET)
Caller function	7. Caller tear-down (pop callee's return value and arguments from stack)

Recursive linked list traversal

Problem statement: Convert the following function from C to LC-3. This function recursively traverses a linked list and prints its content.

```
int print_list(node *head)
{
 if (!head) return 0;
 printf("%c", head->data);
return print_list(head->next);
}
```

```
typedef struct tag
{
char data;
struct tag *next;
} node; */
```

```
1  ; data.asm
2
3        .ORIG x2000
4
5        .FILL x43
6        .FILL x2006
7
8        .FILL x41
9        .FILL x2000
10
11       .FILL x46
12       .FILL x2002
13
14       .FILL x45
15       .FILL x0
16
17       .END
```

Main function: (print_list.asm)

```
    .ORIG x3000
MAIN
    LD R5, RSTACK
    LD R6, RSTACK

    LD R0, HEAD
    STR R0, R6, #0 ; push list head address to the stack

    JSR PRINT_LIST

    HALT
HEAD
    .FILL x2004
RSTACK
    .FILL x7000
```

```
PRINT_LIST
    ; Bookkeeping
    ADD R6, R6, #-3 ; Space for bookkeeping
    STR R7, R6, #1  ; Save return address
    STR R5, R6, #0  ; Save prev. frame pointer

    ADD R5, R6, #-1 ; Move frame pointer

    ; if (!head) return 0;
    LDR R1, R5, #4  ; R1 <- head
    BRz DONE        ; if head is NULL

    ; printf("%c", head->data);
    LDR R0, R5, #4
    LDR R0, R0, #0
    OUT
```

```
    ; print_list(head->next)
    LDR R1, R1, #1  ; R1 <- head->next
    ADD R6, R6, #-1 ; Push head->next as parameter
    STR R1, R6, #0

    JSR PRINT_LIST

    ; return
    LDR R0, R6, #0  ; Load return value to R0
    STR R0, R5, #3  ; Store return value from R0 to correct location

    ADD R6, R6, #2

    BR TEARDOWN

DONE
    AND R0, R0, #0
    STR R0, R5, #3

TEARDOWN
     LDR R7, R5, #2  ; Restore R7
     LDR R5, R5, #1  ; Restore R5
     ADD R6, R6, #2  ; Pop stack
     RET
     .END
```

ILLINOIS

Data file: data.asm

```
1  ; data.asm
2
3      .ORIG x2000
4
5      .FILL x43
6      .FILL x2006
7
8      .FILL x41
9      .FILL x2000
10
11     .FILL x46
12     .FILL x2002
13
14     .FILL x45
15     .FILL x0
16
17     .END
```