Extra review session

CSE 30: Computer Organization and Systems Programming

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
ListNode * reverse(ListNode *aNode) {
    if (aNode == NULL) return aNode;
    ListNode * tmp = aNode->next;
    aNode->next = aNode->prev;
    aNode->prev = tmp;
    if (aNode->prev == NULL)
        return aNode;
    return reverse(aNode->prev);
}

Translated ARM code:

```
reverse: push {r4, r11, ip, lr}
    cmp r0, #0
    beq ret
    ldr r1, [r0, #4] @ r1 = aNode->next
    ldr r1, [r0, #8] @ r2 = aNode->prev
    str r2, [r0, #4] @ aNode->next = r2
    str r1, [r0, #8] @ aNode->prev = aNode->next
    cmp r1, #0 @ r1 is the new aNode->prev
    beq ret
    mov r0, r1
    bl reverse ]
```

- If this happens, the push & the pop may be eliminated.
- Optimizing stackspace
- The compiler may optimize this to a loop
- Implementation (also called tail recursion)
C Pointers and Strings

```c
char name1[] = "frank";
char *name2 = "frank";

sizeof(name1) = ?  6 bytes
sizeof(name2) = ?  4 bytes

What is the difference between the memory diagram of name1 & name2?

Name1  ⇒  The base address is not explicitly stored in memory

name2 (is stored in memory)

*name1  "f"
*name2  "f"
```
Memory maps and pointers

- `char * arr[3];`

Using the memory map on the left draw a pointer diagram for the multi-level array "arr".

What is `arr+1`? `0x2004`

`(arr+1) = 0x4100`
Memory maps and pointers

- Assume `p` is mapped to `r0`. Identify dangling pointers:

  1. `p`
  2. `p + 1`
  3. `*p`
  4. `*(p + 1)`

```c
int ***p;
int (**p)[2];
```

What is?
1. `p` 0x2000
2. `p + 1` 0x2004
3. `*p` 0x4100
4. `*(p + 1)` 0x4200
Memory maps and pointers

- Assume p is mapped to r0. Identify dangling pointers

\[
\begin{align*}
\text{int } &\ast (\ast p)[2]; \\
\text{int } &\ast (\ast p)[2][3];
\end{align*}
\]

What is \( p+1 \) in each case?
Memory maps and pointers

- Assume p is mapped to r0. Identify dangling pointers

```c
int **p[2];
```

```c
int (*p[2])[3];
```

What is p+1 in each case?
int compute_FIR( int* h, int* z, int ntaps, int input){
    int ii;
    int accum;
    z[0] = input;
    accum = h[ntaps - 1] * z[ntaps - 1];
    for (ii = ntaps - 2; ii >= 0; ii--){
        accum += h[ii] * z[ii];
        z[ii + 1] = z[ii];
    }
    return accum;
}
Number representation

- Unsigned
- Sign and magnitude
- 1’s complement
- 2’s complement

**Most positive (4-bit)**

- Unsigned: 1111 (15)
- Sign and magnitude: 0111 (7)
- 1’s complement: 0111 (7)
- 2’s complement: 0111 (7)

**Most negative (4-bit)**

- Signed: 1111 (−7)
- Sign and magnitude: 1000 (−7)
- 2’s complement: 1000 (−8)

**Condition for overflow:**

- $Z = 0$
- $V = 1$
- $C = 1$
- $N = 0$
Bitwise and Logical

- a: 1100 0011, b: 0011 1100
- Write the result of the following expressions
  - a == b
  - a && b
  - a & b
  - a || b
  - a | b
  - a != b
  - ~a
  - a^b

int x = 10, *p;
if (p && (*p == 10)) /* Would this result in a segfault? */
Exam Location

- Section A: Wed (12/9) 8am to 11am, WLH 2001
- Section B: Thur (12/10) 11:30a to 2:30a, REC Gym
- Section C: Fri (12/11) 3pm to 6pm, CENTER 119 and CENTER 214
- (Seating chart will be posted on Piazza)
GOOD LUCK FOR THE FINAL!