Recursion

CSE 30: Computer Organization and Systems Programming

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
Recursion in ARM

```assembly
Recursion in ARM

fact:
    CMP r0, #1
    BLE ret_one
    MOV r1, r0
    SUB r0, r0, #1
    BL fact
    MUL r0, r0, r1
    B end

ret_one: MOV r0, #1
end:
    BX lr
```

```c
int fact (int n)
{
    if (n <= 1)
        return 1;
    else
        return n * fact(n - 1);
}
```

What is the value returned by fact(1)?

A. One  
B. Two  
C. Three  
D. Six  
E. None of the above
Recursion in ARM

fact:

CMP r0, #1
BLE ret_one
MOV r1, r0
SUB r0, r0, #1
BL fact
MUL r0, r0, r1
B end

ret_one: MOV r0, #1
end:
BX lr

int fact (int n)
{
    if (n <= 1)
        return 1;
    else
        return n * fact(n - 1);
}

What is the value returned by fact(2)?
A. One
B. Two
C. Four
D. Six
E. None of the above
Recursion in ARM

\textbf{fact:} \texttt{push \{lr\}}
\begin{align*}
\text{CMP} & \ r0, \ #1 \\
\text{BLE} & \ \text{ret\_one} \\
\text{MOV} & \ r1, \ r0 \\
\text{SUB} & \ r0, \ r0, \ #1 \\
\text{BL} & \ \text{fact} \\
\text{MUL} & \ r0, \ r0, \ r1 \\
\text{B} & \ \text{end}
\end{align*}

\texttt{ret\_one:} \texttt{MOV} \ r0, \ #1
\begin{align*}
\text{end:} & \texttt{pop \{lr\}} \\
\text{BX} & \ lr
\end{align*}

\begin{verbatim}
int fact (int n) {
  if (n <= 1) return 1;
  else return n * fact(n - 1);
}
\end{verbatim}

What is the value returned by fact(3)?
\begin{enumerate}
  \item A. One
  \item B. Two
  \item C. Four
  \item D. Six
  \item E. None of the above
\end{enumerate}
Recursion in ARM

```assembly
fact:  push {r1, lr}
      CMP r0, #1
      BLE ret_one
      MOV r1, r0
      SUB r0, r0, #1
      BL fact
      MUL r0, r0, r1
end:

ret_one: MOV r0, #1
end:      pop {r1, lr}
BX lr
```

int fact (int n) {
    if (n <= 1)
        return 1;
    else
        return n * fact(n - 1);
}

What is the value returned by fact(3)?
A. One  
B. Two  
C. Four  
D. Six  
E. None of the above
void foo ( ) { 
    int arr[3];
    int  j=0;
    for (j=0; j<=3; j++)
    *(arr+j)=0;
}

When the function foo () is called, the program gets stuck in an infinite loop.

• Explain the reason for this behavior.
• Which of the following represents the relative location of the elements of the array and the variable j in memory?
  Assume memory locations increase left to right
  
D.  Either A or B
E.  None of the above
void foo () {
    char arr[3];
    int j=0;
    for (j=0; j<=3; j++)
        *(arr+j)=0;
}

Do you think the program would still get stuck in an infinite loop if arr was a char array?
Assume data is aligned and byte ordering is little endian.
A. Yes
B. No
Explaining weird program behavior

```c
void foo ( ) {
    char arr[4];
    int j=0;
    for (j=0; j<=4; j++)
        *(arr+j)=0;
}
```

Do you think the program would still get stuck in an infinite loop for the new code? Assume data is aligned, byte ordering is Little Endian.
Think about why or why not
A. Yes
B. No
Multidimensional arrays: 2D

- **Declaration**

```c
int a[3][4]; /* Conceptually 2D matrix with 3 rows 4 columns */
```

<table>
<thead>
<tr>
<th>Row</th>
<th>Column 0</th>
<th>Column 1</th>
<th>Column 2</th>
<th>Column 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 0</td>
<td><code>a[0][0]</code></td>
<td><code>a[0][1]</code></td>
<td><code>a[0][2]</code></td>
<td><code>a[0][3]</code></td>
</tr>
<tr>
<td>Row 1</td>
<td><code>a[1][0]</code></td>
<td><code>a[1][1]</code></td>
<td><code>a[1][2]</code></td>
<td><code>a[1][3]</code></td>
</tr>
</tbody>
</table>

- Element in row i, column j is retrieved as `a[i][j]`
- ‘a’ is a pointer to an integer array of size 4
int arr[3][4]; /*Conceptually 2D matrix with 3 rows 4 columns */

In memory the elements of ‘arr’ are stored in a contiguous memory block

Row 0 elements  Row 1 elements  Row 2 elements
(arr+i) increments the address of ‘arr’ by how many bytes?

A. \( i \times \text{sizeof(int)} \)

B. \( i \times \text{sizeof(int*)} \)

C. \( i \times \text{sizeof(int)} \times \text{number of columns} \)
Express \( \text{arr}[i][j] \) using ‘\text{arr}’ as a pointer

A. \(*((\text{arr}+i)+j)\)
B. \(*(*(\text{arr}+i)+j)\)
C. \(*((\text{arr}+4*i*\text{sizeof(int)})+j*\text{sizeof(int)})\)
D. None of the above