Recursive Control Flow

ECE 362
https://engineering.purdue.edu/ee362/

© Rick
Consider a recursive subroutine:

```c
int total = 0;

int sum(int x) {
    if (x > 0) {
        total += x;
        sum(x-1);
    }
}

int main() {
    sum(3);
}
```

```assembly
Consider a recursive subroutine:

int total = 0;

int sum(int x) {
    if (x > 0) {
        total += x;
        sum(x-1);
    }
}

int main() {
    sum(3);
}
```
A different way to look at it.

main:
    movs r0, #3
    bl sum
    wfi

sum:
    push {lr}
    if1:
        cmp r0, #0
        ble endif1
        ldr r1, =total
        ldr r2, [r1]
        adds r2, r0
        str r2, [r1]
        subs r0, #1
        bl sum
    endif1:
    nop
    pop {pc}

main:
    movs r0, #3
    bl sum
    wfi

(stack)

SP
A different way to look at it.
A different way to look at it.

```
main:
movs r0, #3
bl sum
wfi

sum:
push {lr}
if1:
cmp r0,#0
ble endif1
ldr r1,=total
ldr r2,[r1]
adds r2,r0
str r2,[r1]
subs r0,#1
bl sum
endif1:
nop
pop {pc}

main:
movs r0, #3
bl sum
wfi

wfi
```

(stack)

<table>
<thead>
<tr>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>wfi</td>
</tr>
</tbody>
</table>
A different way to look at it.
A different way to look at it.
A different way to look at it.
A different way to look at it.
A different way to look at it.
A different way to look at it.

```
main:
movs r0, #3
bl sum
wfi

sum: push {lr}
if1: cmp r0,#0
    ble endif1
    ldr r1,=total
    ldr r2,[r1]
    adds r2,r0
    str r2,[r1]
    subs r0,#1
    bl sum
endif1: nop
    pop {pc}

main:
movs r0, #3
bl sum
wfi

(stack)  wfi  nop  nop  nop  SP
```
A different way to look at it.
A different way to look at it.
A different way to look at it.

```assembly
main:
    movs r0, #3
    bl sum
    wfi

if1:
    cmp r0, #0
    ble endif1
lad r1,=total
    ldr r2,[r1]
    adds r2,r0
    str r2,[r1]
    subs r0,#1
    bl sum

endif1:
    nop
    pop {pc}

main:
    movs r0, #3
    bl sum
    wfi
```

(stack)   wfi   nop   nop   nop
A different way to look at it.

```assembly
main:
    movs r0, #3
    bl  sum
    wfi

if1:
    cmp r0,#0
    ble endif1
    ldr r1,=total
    ldr r2,[r1]
    adds r2,r0
    str r2,[r1]
    subs r0,#1
    bl  sum
endif1:
    nop
    pop {pc}

main:
    movs r0, #3
    bl  sum
    wfi
```

(stack)

<table>
<thead>
<tr>
<th></th>
<th>wfi</th>
<th>nop</th>
<th>nop</th>
<th>nop</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A different way to look at it.

```
main:
    movs r0, #3
    bl   sum
    wfi

if1:
    cmp  r0,#0
    ble  endif1
    ldr  r1,=total
    ldr  r2,[r1]
    adds r2,r0
    str  r2,[r1]
    subs r0,#1
    bl   sum

endif1:
    nop
    pop  {pc}

main:
    movs r0, #3
    bl   sum
    wfi
```

(stack)  wfi  nop  nop  nop