CEG2400 - Microcomputer Systems

Lecture 6: Higher Level Constructs

Philip Leong

Introduction

- ARM instructions and simple prorgams introduced
- Talk about assembly language programming, in particular
  - Expressions
  - Conditionals
  - Loops

Pointers

- int *p; p = p + 1;
  - Since p points to a 4-byte value, p must be incremented by 4
  - If p is in register r0 “add r0,r0,#4”
- int *p; p = p + t;
  - p must be incremented by 4 * t so if t is in r1
  - add r0,r0,r1,lsl #2

Arrays

- Same as pointers since a[i] is the same as *(a + i)

Conditionals

- if (a > b) c = a; else c = b;
- a, b and c are in r0, r1, r2

```
   cmp r0,r1
   ble l1
   mov r2,r0
   b end
l1 mov r2,r1
end
```


Conditionals

- Method in previous slide most general, can have many statements in the “if” and “else” parts of the condition
- For simple cases such as the example, more efficient to use conditional execution

```
cmp r0,r1
movgt r2,r0
movle r2,r1
```

Switches

```
switch (e) {
    case 0: stmts0;
    case 1: stmts1;
    …
    default: stmtsd;
}
```

- Can be handled using “if” statements

```
tmp = e;
if (tmp == 0) {
    stmts0;
}
else if (tmp == 1) {
    stmts1;
}
…
else {
    stmtsd;
}
```


Jump table

```
    adr r1, jtable          ; get base of jump table
    cmp r0,#jtablemax       ; check for overrun
    ldrls pc,[r1,r0,lsl #2] ; jump to stmts
    ; stmtd here
    b exit
L1 ; stmt1 here
    b exit
L2 ; stmt2 here
    b exit

jtable dcd L1
    dcd L2
jtableend
```


For loop

- for (i = 0; i < 10; i++)
  a[i]=0;

```
    mov r1,#0
    adr r2,a            ; pointer to a[]
    mov r0,#0
L   cmp r0,#10
    bge exit
    str r1,[r2,r0,lsl #2]   ; a[i] = 0
    add r0, r0, #1          ; i++
    b L
exit
; This code could be optimised in a number of ways, please think about
    it
```


While

```
while (expr)
  body;

LOOP    …           ; expr
        beq  exit
        …           ; body
        b LOOP
EXIT
```


More efficient while

```
while (expr)
  body;

        b TEST
LOOP    …           ; body
TEST    …           ; expr
        bne LOOP
EXIT
```


Do while

```
do
{
  body;
} while (expr);

LOOP    …           ; body
        …           ; expr
        bne LOOP
EXIT
```