

Exercises ARM Architecture and Instruction Set

Ingo Sander
ingo@imit.kth.se



ARM assembly language if-then-else statement



- How is the following C statement implemented in ARM assembly language?

```
If (x > 5) then
{
...
/* Ifcode */
...}
Else
{
...
/* Elsecode */
...}
```

```
X EQU 0x00008000
...
ADR r1, x
LDR r2, [r1]
CMP r2, #5
BLE Else
If ...
... ; Ifcode
B Endif
Else ...
... ; Elsecode
Endif ...
```

ARM assembly language while statement



- How is the following C statement implemented in ARM assembly language?

```
While (x < 3)
{
/* Whilecode */
}
```

```
X EQU 0x00008000
...
While ADR r1, x
LDR r2, [r1]
CMP r2, #3
BGE Endwhile
... ; Whilecode
B While
Endwhile ...
```

ARM assembly language for statement



- How is the following C statement implemented in ARM assembly language?

```
For (x=1; x < 10; x++)
{
/* Forcode */
}
```

```
Init MOV r2, #1
For CMP r2, #10
BGE Endfor
... ; Forcode
ADD r2, r2, #1
B For
Endfor ...
```

ARM assembly language switch statement



- How is the following C statement implemented in ARM assembly language?

```
switch (test) {
    case 0: ... break;
    case 1: ... }
```

```
ADR r2, test
LDR r0, [r2]
ADR r1, switchtab
LDR r15, [r1, r0, LSL #2]
...
switchtab
    DCD ... ; Code for case0
    DCD ... ; Code for case1
```

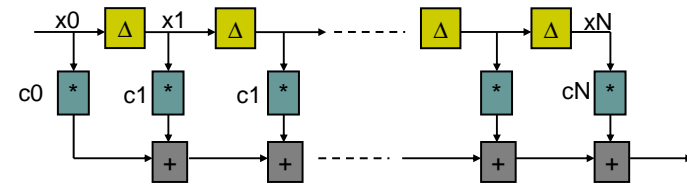
- LDR r15, [r1, r0, LSL #2] is the core of the switch statement. LSL #2 multiplies the index by 4 since the address space for a word is 4 bits.
- Then the new address is calculated and the PC is loaded with that address!

ARM assembly language FIR-Filter



- How is the following C code for an FIR-filter implemented in ARM assembly language?

```
for (i=0, f=0; i<N; i++)
    f = f + c[i]*x[i];
```



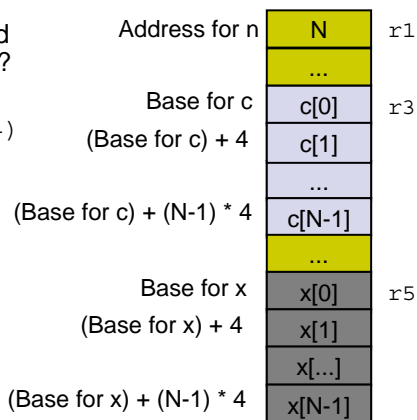
ARM assembly language FIR-Filter



- How is the following C code for an FIR-filter implemented in ARM assembly language?

```
for (i=0, f=0; i<N; i++)
    f = f + c[i]*x[i];
```

```
r0 : i
r2 : f
r8 : i * 4
```



ARM assembly language FIR-Filter



```
        AREA  Constants, DATA
N       DCD   4
C       DCD   1, 0, -1, 1
X       DCD   1,2,3,4
...
; loop initiation code (Code Block)
MOV r0,#0    ; use r0 for I
MOV r8,#0    ; use separate index for array addresses
ADR r2,N     ; get address for N
LDR r1,[r2]  ; get value of N
MOV r2,#0    ; use r2 for f
ADR r3,c     ; load r3 with base of c
ADR r5,x     ; load r5 with base of x
```

ARM assembly language

FIR-Filter



```
; loop body
loop
    LDR r4,[r3,r8] ; get c[i]
    LDR r6,[r5,r8] ; get x[i]
    MUL r7,r4,r6   ; compute c[i]*x[i]
    ADD r2,r2,r7   ; add into running sum
    ADD r8,r8,#4   ; add one word offset to array
                    ; address
    ADD r0,r0,#1   ; add 1 to i
    CMP r0,r1     ; exit?
    BLT loop      ; if i < N, continue
```

ARM assembly language

Improved FIR-Filter



```
                AREA Constants, DATA
N                DCD 4
C                DCD 1, 0, -1, 1
X                DCD 1,2,3,4
...
; loop initiation code (Code Block)
MOV r0,#0       ; use r0 for I
; MOV r8,#0     ; not needed
ADR r2,N        ; get address for N
LDR r1,[r2]     ; get value of N
MOV r2,#0       ; use r2 for f
ADR r3,c        ; load r3 with base of c
ADR r5,x        ; load r5 with base of x
```

ARM assembly language

Improved FIR-Filter



```
; loop body
loop
; LDR r4,[r3,r8] ; modified
; LDR r6,[r5,r8] ; modified
    LDR r4,[r3], #4 ; get c[i],increment address by 4
    LDR r6,[r5], #4 ; get x[i],increment address by 4
; MUL r7,r4,r6   ; modified
; ADD r2,r2,r7   ; modified
; ADD r8,r8,#4   ; not needed

    MLA r2, r4, r6, r2 ; compute c[i] * x [i] + f
    ADD r0,r0,#1     ; add 1 to i
    CMP r0,r1       ; exit?
    BLT loop        ; if i < N, continue
```

Exercise 2.5 b



- Write ARM Assembly code to implement the following C assignment:

$$y = (c-d) + (e-f)$$

Assumption:

- a, b, c, d, e, f are stored in an array at label numbers

```
                AREA my_area, DATA
numbers        DCD 1,2,3,4,5,6
```



Exercise 2.5b

```

;; y = (c-d) + (e-f)
LDR r9, =numbers      ; Read Start Address for
                       ; Number Block

LDR r1, [r9, #8]!     ; C is saved in r1
                       ; (preindex with writeback)

LDR r2, [r9, #4]!     ; D is saved in r2
LDR r3, [r9, #4]!     ; E is saved in r3
LDR r4, [r9, #4]!     ; F is saved in r4
SUB r5, r1, r2        ; r5 = C - D
                       ; (with update of CPSR)

SUB r6, r3, r4        ; r6 = E - F
ADD r7, r5, r6        ; Y = r5 + r6
STR r7, [r9, #8]!    ; Y is stored after X

```



Exercise 2.9

- Write ARM assembly code to implement the following C conditional:

```

if (x - y < 3) {
    a = b - c;
    x = 0; }
else {
    y = 0;
    d = e + f + g; }

```

- Assumption:

a, b, c, d, e, f, g, x, y are stored as array at label q29



Exercise 2.9

```

LDR r10, =q29
LDMIA r10!, {r1-r9}
SUB r11, r8, r9
CMP r11, #3
BGE else
if SUB r1, r2, r3
   MOV r8, #0
   B endif
Else MOV r9, #0
     ADD r12, r5, r6
     ADD r4, r12, r7
endif B endif

```