Tutorial2.2

Assembly language

—loops , if, arithmetic opcode

Lei Wang

UNIVERSITY OF CALGARY

Arithmetic Instructions

- Addition : add
  - Register(64-bit and 32-bit):
    - Eg: add x19, x20, x21 // x19 = x20 + x21
    - Eg: add w19, w19, w20 // w19 = w19 + w20
  - Immediate(64-bit and 32-bit):
    - Eg: add x20, x20, 1 //x20 = x20 + 1
    - Eg: add w27, w19, 4 //w27 = w19 + 4
- Substraction: sub
- Multiplication: mul wd, wn, wm => wd = wn * wm
- Multiply-Add: madd wd, wn, wm wa => wd = wa + (wn * wm)
- Multiply-Substract: msub wd, wn, wa => wd = wa – (wn * wm)

Arithmetic Instructions

- Multiply-Negate: mneg wd, wn, wm => wd = -(wn*wm)
- Division
  - Signed form: sdiv wd, wn, wm => wd = wn/wm
  - Unsigned form: udiv w0, w1, w2 => w0 = w1/w2
  - Dividing by 0 does not generate an exception

Loops

- Branch instructions
  - b label => goto in the c language
- Condition codes
  - cmp Xn, Xm compares the value of xn and xm, the results in this instruction have 6 flags
- Conditional branch: depend the condition flag to make a branch
  - b.eg: equal then branch ==
  - b.ne: not equal then branch !=
  - b.gt: greater than then branch >
  - b.ge: greater than or equal then branch >=
  - b.lt: less than then branch <
  - b.le: less than or equal then branch <=

Loops

- While loop
  - Post-test loop: the loop body will be executed at least once

Eg:

```
        define(x_r, x19)
        mov     x_r, 1
top:    statement forming
        loop body
        add     x_r, x_r, 1
        cmp     x_r, 10
        b.le    top
```

Loops

- Pre-test loop: the loop body will possibly not be executed

eg:

```
define(x_r, x19)
        mov     x_r, 0
test:   cmp     x_r, 10
        b.ge    done
        statements forming
        body of loop
        add     x_r, x_r, 1
        b       test
done:   statement following loop
```

Loops

- Optimization

Eg:

```
Define(x_r, x19)
        mov     x_r, 0
        b       test
top:    statementsforming
        body of loops
        add     x_r, x_r, 1
test:   cmp     x_r, 10
        b.lt    top
        statement following loop
```

If / else

- Use branching
  - Eg:

```
        define(a_r, x19)
        define(b_r, x20)
        define(c_r, x21)
        define(d_r, x22)
        cmp     a_r, b_r
        b.le    else
        add     c_r, a_r, b_r
        add     d_r, c_r, 5
        b       next
else:   sub     c_r, a_r, b_r
        sub     d_r, c_r, 5
next:   statement after if-else construct
```

example—pre-test loop

- Pre-test while

```
fmt: .string "Loop: %d\n"


      .balign 4
      .global main
main: stp x29, x30, [sp, -16]!
      mov x29, sp

      mov x19, 0

test: cmp  x19, 10
      b.ge done


      adrp x0, fmt
      add  x0, x0, :lo12:fmt
      add  x1, x19, 1

      bl   printf

      add  x19, x19, 1
      b    test

done: mov  w0, 0

      ldp  x29, x30, [sp], 16
      ret
```

example—optimization

- Using macros and optimized loop

```
define(loop_r, x19)
define(fp, x29)
define(lr, x30)

fmt: .string "Loop: %d\n"

        .balign 4
        .global main

main: stp  fp, lr, [sp, -16]!
      mov  fp, sp

      mov  loop_r, 0
      b test

top:
      adrp x0, fmt
      add  x0, x0, :lo12:fmt
      add  x1, loop_r, 1
      bl   printf

      add  loop_r, loop_r, 1
test: cmp  loop_r, 10
      b.lt top

done: mov  w0, 0

      ldp  fp, lr, [sp], 16
      ret
```

script—record your screen

- SYNOPSIS
  - script [-a] [filename]
- DESCRIPTION
  - script makes a record of everything printed on your screen. The record is written to filename.
  - If no file name is given, the record is saved in the file 'typescript'.
  - The script ends when exit is typed or when Ctrl+D is pressed.
- OPTIONS
  - -a Append the session record to filename, rather than overwrite it.
- NOTES
  - script places everything that appears on the screen in filename, including prompts.

script—record your screen

- Use script to record your debugging info
- ***script script1.txt*** *// start script logging, saving or outputting the log to a file called script1.txt*
- ***gdb exacutablefilename*** *// start debugging your program*
- ***display/i $pc*** *// display the current instruction at every break*
- ***b topoftheloop*** *// break at the start of your loop*
- ***r*** *// run your program, it will pause at the first breakpoint*
- ***ni*** *// start single stepping through the loop...*
- ***p $reg*** *// print all the important registers*
- ***ni*** *// print the register a few times as you step through your program with ni, to show it being modified*
- ***ni***
  ... *// ...til end of loop*
- ***c*** *// continue until end of program*
- ***q*** *// quit gdb*
- ***exit*** *// exit script, stop logging*

scp—download file from the remote server

- Syntax
- scp [username]@[hostname]:[remote source file] [local destination]

Eg:

scp lei.wang2@arm.cpsc.ucalgary.ca:/home/grads/lei.wang2/tutorial2/expr.s G:/serverfile/

```
PS C:\Users\16568> scp lei.wang2@arm.cpsc.ucalgary.ca:/home/grads/lei.wang2/tutorial2/expr.s G:/server_file
lei.wang2@arm.cpsc.ucalgary.ca's password:
expr.s                                        100%  787     0.8KB/s   00:00
```

exercise

- Work on your assignment 1
- You are expected to submit 4 files:
  - assign1a.s
  - assign1b.asm
  - script1.txt
  - Script2.txt
- Don’t forget to comment each line of assembly code