Tutorial 6.3

command line arguments & external pointer arrays

Lei Wang

lei.wang2@ucalgary.ca

External Arrays of Pointers

- A string literal is a read-only array of characters, allocated in the text section
- The literal usually has a label, which represents the address of the first character in the array
- External arrays of pointers are created with a list of labels

```
        .text
fmt:    .string     "hello world"
```

label string

External Arrays of Pointers

```c
#include <stdio.h>

char *season[4] = {"spring", "summer",
"fall", "winter"};

int main()
{
        register int i;
        for (i = 0; i < 4; i++){
                printf("season[%d] = %s\n", i,
season[i]);
        }

        return 0;
}
```

```
define(i_r, w19)
define(base_r, x20)
            .text
fmt:        .string             "season[%d] = =%s\n"
spr_m:      .string             "spring"
sum_m:      .string             "summer"
fal_m:      .string             "fall"
win_m:      .string             "winter"

            .data
            .balign   8
season:     .dword    spr_m, sum_m, fal_m, win_m

            .text
            .balign 4
            .global main
main:       stp       x29, x30, [sp, -16]!
            mov       x29, sp
            mov       i_r, 0
            b         test
top:        adrp      x0, fmt
            add       x0, x0, :lo12:fmt
            mov       w1, i_r
            adrp      base_r, season_m
            add       base_r, base_r, :lo12:season_m
            ldr       x2, [base_r, i_r, SXTW 3]
            bl        printf
            add       i_4, i_r, 1
test:       cmp       i_r, 4
            b.lt      top
            ldp       x29, x30, [sp], 16
            ret
```

Command-line arguments

- pass values from the shell into your program
  - argc: the number of arguments
  - argv[]: an array of pointers to the arguments
  - argc is in w0 and argv[] is in x1

```
int main(int argc, char *argv[])
{
        register int i;
        for(i = 0; i < argc; i++){
                printf("%s\n", argv[i]);
        }
        return 0;
}
```

```
define(i_r, w19)
define(argc_r, w20)
define(argv, x21)

fmt:    .string "%s\n"

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

        mov     argc_r, w0
        mov     argv_r, x1

        mov     i_r, 0
        b       test
top:    adrp    x0, fmt
        add     x0, x0, :l012:fmt
        ldr     x1, [argv_r, i_r, SXTW 3]
        bl      printf

        add     i_r, i_r, 1
test:   cmp     i_r, argc_r
        b.lt    top
        ldp     x29, x30, [sp], 16
        ret
```

coding practice

- input a string using command-line argument and convert it to integer using atoi function

```
// Converts a number from ASCII string to int
// Usage: ./a.out 12345
// Output: "The number you entered is: 12345"

        .text
fmt:    .string "The number you entered is: %d\n" // format takes an int variable

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

        mov w19, 1                              // the 1st arg (index 0) is the program name
                                                // we want the 2nd arg, the number entered

        ldr x0, [x1, w19, SXTW 3]               // x1 is the base address to the external pointer
                                                // array containing pointers to all our args
                                                // w19 (1) is the index, SXTW 3 to calculate offset
        bl atoi                                 // convert ASCII string to integer, result in w0

        mov w1, w0                              // set up 2nd arg for printf
        adrp x0, fmt                            // set up 1st arg for printf
        add x0, x0, :lo12:fmt
        bl printf                               // print the number

        mov w0, 0
        ldp x29, x30, [sp], 16
        ret
```

reference

- lecture slides from cpsc 355