Tutorial 5.4

assignment 4

Lei Wang

lei.wang2@ucalgary.ca

UNIVERSITY OF CALGARY

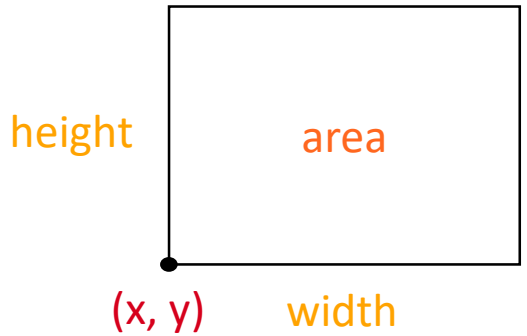
Structure definition and initialization

```
struct point{
        int x, y;
};
//point_size = 8
struct dimension{
        int width, height;
}
//dimension_size = 8
struct box{
        struct point origin;
        struct dimension size;
        int area;
};
//box is a nested structure with size: 8 + 8 + 4 = 20

struct box newBox()
{
        struct box b;
        b.origin.x = 0;
        b.origin.y = 0;
        b.size.width = 1;
        b.size.height = 1;
        b.area = b.size.width * b.size.height;

        return b;
};
```

point: (x, y)

dimension: (width, height)

box: (point, dimension, area)



UNIVERSITY OF CALGARY

Structure modify and print

```
void move(struct box *b, int deltaX, int deltaY)
{
        b->origin.x += deltaX;
        b->origin.y += deltaY;
}

void expand(struct box *b, int factor)
{
        b->size.width *= factor;
        b->size.height *= factor;
}

void printBox(char *name, struct box *b)
{
       printf("Box %s origin = (%d, %d) width = %d height = %d area = %d\n",
              name, b->origin.x, b->origin.y, b->size.width, b->size.height,
              b->area);
}
```

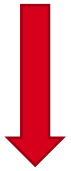
Use addr of string as argument to print

```c
#include <stdio.h>

int main(){
        char * s = "abcdefg";
        printf("the string is: %s", s);
}
```

```
fmt0:   .string "the string is: %s"
fmt1:   .string "abcdefg"

        .balign 4
        .global main
main:   stp      x29, x30, [sp, -16]!
        mov     x29, sp
        // set the first arg
        adrp    x0, fmt0
        add     x0, x0, :lo12:fmt0
        // set the second arg
        adrp    x1, fmt1
        add     x1, x1, :lo12:fmt1

        bl      printf

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

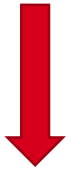
Compare two structure

```
int equal(struct box *b1, struct box *b2)
{
        int result = FALSE;
        if (b1 -> origin.x == b2 -> origin.x){
                if (b1 -> origin.y == b2 -> origin.y){
                        if (b1 -> size.width == b2 -> size.width){
                                if (b1 -> size.height == b2 -> size.height)
                                        result = TRUE;
                        }
                }
        }
        return result;
}
```



equivalent

```
if((b1->origin.x==b2->origin.x)&(b1->origin.y==b2->origin.y)&(b1-> size.width==b2->size.width)&(b1->size.height==b2->size.height))
  {
          result = TRUE;
  }
```

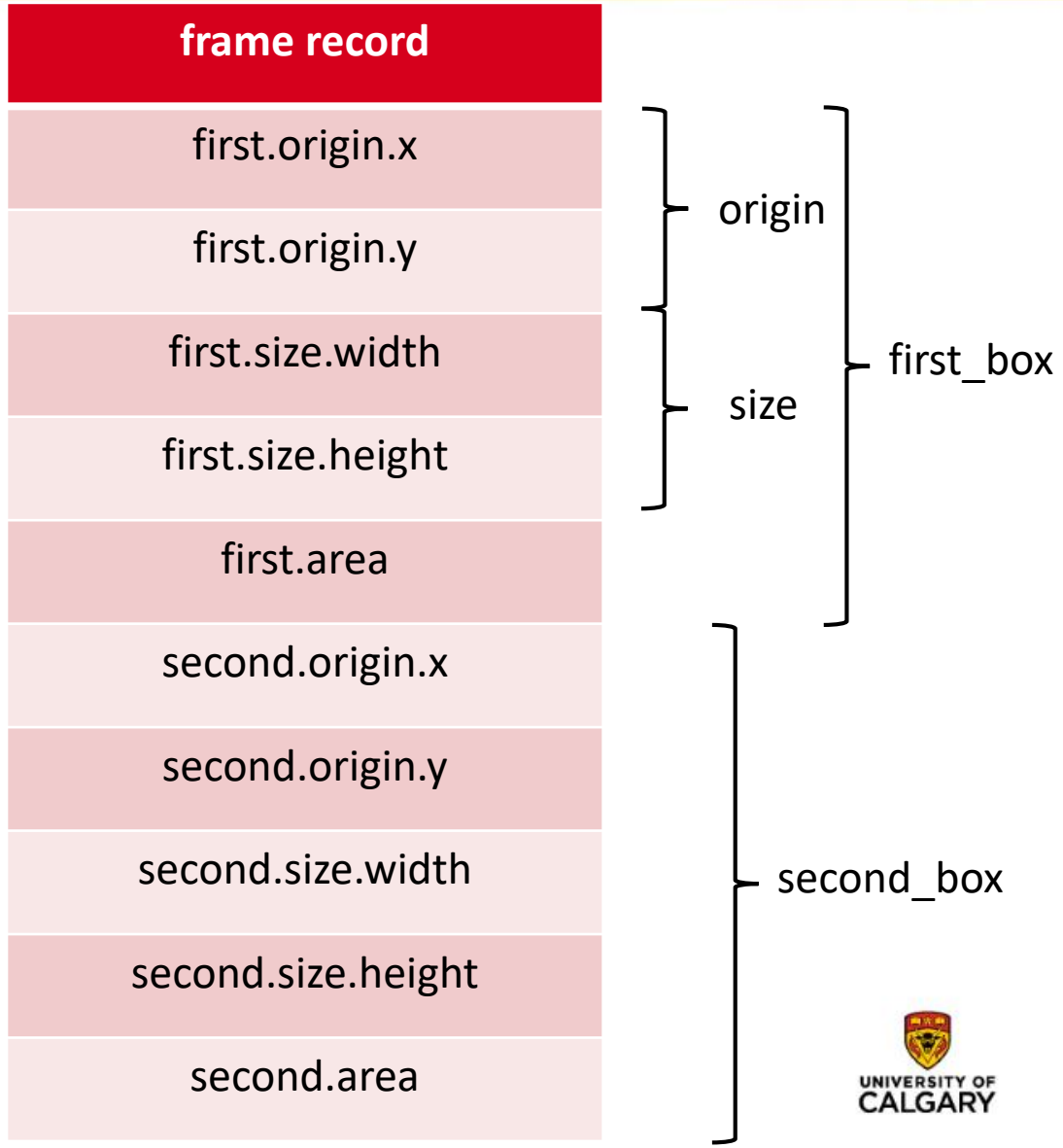


equivalent

in assembly if any of these conditions doesn't meet then jump out to return

main

```
int main()
{
        struct box first, second;

        first = newBox();
        second = newBox();

        printf("Initial box values: \n");
        printBox("first", &first);
        printBox("second", &second);

        if(equal(&first, &second)){
                move(&first, -5, 7);
                expand(&second, 3);
        }

        printf("\n Change box values: \n");
        printBox("first", &first);
        printBox("second", &second);
}
```



How to start? calculate the equates

```
struct point{
        int x, y;
};

struct dimension{
        int width, height;
}

struct box{
        struct point origin;
        struct dimension size;
        int area;
};
```

eg: x_s = 0
y_s = 4
point_size = 8

Tips for assignment 4

- steps to display structure in gdb
  - b to set breakpoint in every functions that set the value of struct
  - r to run the program and stop at the first breakpoint
  - c to continue to next breakpoint
  - x/nd $fp+offset to display the structure
  - c to continue to next breakpoint
  - x/nd $fp+offset to display the structure again
  - …
- use w9-w15/x9-x15 for temporary register in subroutines
- return structure using x8 to store the address of returning memory
- print or modify structure using x0-x7 to pass the address as arg

work period