TDT4205 Compilers
Exercise 4

Stian Hvatum (hvatum)
MTDT

March 3, 2012

Contents

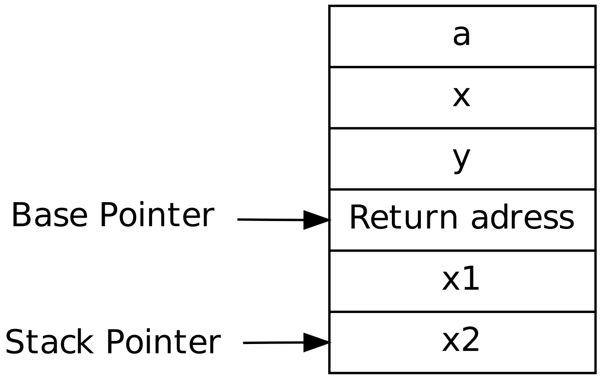
PART 1 Theory and Assembly Programming

Task 1.1 Stack Frames

1. What is a stack frame

A *stack frame* is a location in a programs logical memory, or more precicely, on the *stack*, where it keeps the current local variables. The stack frame grows as local variables are added, and shrinks as they are poped of, eg. if they are not going to be used any more.

2. Stack frame illustration



3. Setting up and tearing down stack frames

Task 1.2 x86 Assembly Programming

The complete file *foo.s* is attached with the delivery of this file.

```
foo:
    /* Store old base pointer on top of stack */
    pushl   %ebp

    /* Set new stack base (ebp) to old top-of-stack (esp) */
    movl    %esp, %ebp

    /* Store 0 in ecx  (loop starts at 1, but is incremented in first test) */
    movl    $0,   %ecx

    /* Store 0 on the stack, our sum value */
    pushl   $0

    /* And start loop-test */
    jmp tst_lp

lbody:
    /* Loop body */
    /* Modulo ← divide and check rest-register */


    /* Check for input divisible by 3 */
    movl    %ecx, %eax
    movl    $3,   %ebx
    cdq
    idiv    %ebx
    /* edx now contains ecx mod 3 */
    cmp     $0,   %edx
    jz tst_ok /* Test true */

     /* Check for input divisible by 5 */
    movl    %ecx, %eax
    movl    $5,   %ebx
    cdq
    idiv    %ebx
    /* edx now contains ebx mod 5 */
    cmp     $0,   %edx
    jnz tst_lp /* Test false */
tst_ok:
    addl    %ecx, -4(%ebp)

tst_lp:
    /* Get the function argument and store in ebx */
    movl    8(%ebp), %ebx

   /* Increment and test */
    inc %ecx
    /* if ebx < ecx   jump to start of loop */
    cmp %ebx, %ecx
    jl lbody

    pushl -4(%ebp)
    /* Print results */
    /* sum is on top of stack */
    pushl   $.STRING0
    call    printf

    /* Clean up on stack */
    addl    $8, %esp
    /* Clean up stack frame */
    leave

    /* Return home */
    ret
```

Task 1.3 Symbol Tables

1. Stack offset

a -4

b -8

c $-28(-8-(4 \cdot 5))$

2. Lexical depth