143A Midterm - 2020 Spring

Q1 Calling Conventions

15 Points

Q1.1

5 Points

Use assembly to create a proper call site for the following C function invocation (i.e., invoke this function with the below arguments)

```
ret = foo(1, 2, 3, 4);
```

where the `foo` function looks like this:

```
int foo (int x, int y, int z, int w) {

    int a = 5, b = 6;

    a += b + x + y + z + w;

    return a;
};
```

Q1.2

5 Points

Use the assembly to save the result returned by the `foo` function above on the stack, i.e.,

```
...
call foo
// add you asm  code here
```

Q1.3

5 Points

Assume that you program uses x86 32bit machine instructions and maintains the stack frame. Draw the call stack (including the arguments passed to `foo`) at the instruction that starts computing this arithmetic expression inside `foo`: `b + x + y + z + w`

Q2 System call interface

10 Points

Write a simple xv6 or Linux program that starts `grep` redirecting its standard input to the `/foobar.txt` file and connecting its standard output to the pipe that connects to the standard input of `wc -l`. Your code does not have to be perfect C, but has to use all system calls correctly (please explain the usage with comments), you can use either xv6 or Linux system calls.

Q3 Segmentation and paging

15 Points

Q3.1

5 Points

Consider the following 32-bit x86 page table setup.

`CR3` holds `0x00000000`.

The Page Directory Page at physical address `0x00000000`:

```
PDE 0: PPN=0x00001, PTE_P, PTE_U, PTE_W
PDE 1: PPN=0x00002, PTE_P, PTE_U, PTE_W
... all other PDEs are zero
```

The Page Table Page at physical address `0x00001000` (which is PPN `0x1`):

```
PTE 0: PPN=0x00003, PTE_P, PTE_U, PTE_W
PTE 1: PPN=0x00004, PTE_P, PTE_U, PTE_W
... all other PTEs are zero
```

The Page Table Page at physical address `0x00002000` (PPN `0x2`):

```
PTE 0: PPN=0x00006, PTE_P, PTE_U, PTE_W
PTE 1: PPN=0x00007, PTE_P, PTE_U, PTE_W
... all other PTEs are zero
```

Which physical address corresponds to the virtual address `0x0` imagine the segment you're using is configured with the base of `0x1000`

Q3.2
10 Points

Construct a page table that maps three pages at virtual addresses `0x8010 0000`, `0x8010 1000`, and `0x8010 2000` to physical addresses `0x0`, `0x10 1000`, and `0x10 2000`. To define the page table you can use the format similar to the one in the question above

Q4 Linkining and loading
10 Points

Imagine you have the following program:

```c
#include <stdio.h>

int y;

static int inc(int a) {
    return a + 1;
}

int dec(int b) {
    return inc(b) - 1;
}

void main () {
    int x;

    x = 5;
    y = dec(x);

    printf("result:%d\n", y);
}
```

Q4.1
5 Points

For each variable in the program explain where and how it is allocated

Q4.2
5 Points

Imagine the program was compiled to be loaded at address `0x0`. Which symbols in the program need to be relocated if you load this program in memory at address `0x10 0000`