16.317: Microprocessor Systems Design I

Summer 2013

Lecture 6: Key Questions

July 26, 2013

1. Describe the issues involved in accessing data in assembly, including the two general factors the compiler must account for.

2. How does a program handle statically allocated data (data allocated at compile time)?

3. How does a program handle data that are dynamically allocated when a function is called?

4. Describe the structure of a typical x86 stack frame.

5. Describe how array accesses are handled.

6. Describe how conditional statements are handled.

7. Describe how loops are handled.

8. Write a subroutine for each of the operations listed on the following pages. Note that:
   - Subroutine arguments are passed on the stack, and can be accessed within the body of the subroutine starting at address EBP+8.
   - At the start of each subroutine:
     i. Save EBP on the stack
     ii. Copy the current value of the stack pointer (ESP) to EBP
     iii. Create space within the stack for each local variable by subtracting the appropriate value from ESP. For example, if your function uses four integer local variables, each of which contains four bytes, subtract 16 from ESP.
     iv. Local variables can then be accessed starting at the address EBP-4.
   - A subroutine's return value is typically stored in EAX.

`int fact(int n)`

Given a single integer argument, `n`, return n! = n × (n – 1) × (n – 2) × … × 1

```
int max(int v1, int v2)
```

Given two integer arguments, return the largest of the two values.

`void swap(int *a, int *b)`

Given two memory addresses, `a` and `b`, swap the contents of those addresses. You may assume `a` and `b` are offsets into the data segment.