

16.317: Microprocessor Systems Design I

Spring 2013

Lecture 20: Solution to Key Questions

March 22, 2013

a. `int fact(int n)`

Given a single integer argument, n , return $n! = n \times (n - 1) \times (n - 2) \times \dots \times 1$

Solution: Here's a C version of the function, followed by the assembly code that implements it:

```
int fact(int n) {  
    int i;  
    int fact = 1;  
  
    for (i = n; i > 1; i--)  
        fact *= i;  
  
    return fact;  
}
```

Assembly code for factorial function:

```
fact      PROC                                ; Start of subroutine
    push   ebp                                ; Save ebp
    mov     ebp, esp                            ; Copy ebp to esp
    sub     esp, 8                             ; Create space for i, fact

; CODE FOR: int fact = 1;
    mov     DWORD PTR -8[ebp], 1                ; fact = 1

; CODE FOR: i = n;
    mov     eax, DWORD PTR 8[ebp]                ; eax = n
    mov     DWORD PTR -4[ebp], eax                ; i = n

; CODE FOR i > 1
L1:
    cmp     DWORD PTR -4[ebp], 1                ; Compare i to 1
    jle     L2                                    ; If i <= 1, exit loop

; CODE FOR: fact *= i;
    mov     eax, DWORD PTR -8[ebp]                ; eax = fact
    imul    eax, DWORD PTR -4[ebp]                ; eax = fact * i
    mov     DWORD PTR -8[ebp], eax                ; fact = eax = fact * i

; CODE FOR: i--;
    mov     eax, DWORD PTR -4[ebp]                ; eax = i
    sub     eax, 1                                ; eax--
    mov     DWORD PTR -4[ebp], eax                ; i = eax = i - 1
    jmp     L1                                    ; Return to loop start

; CODE FOR: return fact;
L2:
    mov     eax, DWORD PTR -8[ebp]                ; Copy fact to eax, which
                                                ; holds return value

; CLEANUP
    mov     esp, ebp                            ; Clear space for i, fact
    pop     ebp                                ; Restore ebp
    ret                                           ; Return from subroutine
fact      ENDP
```

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

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

Solution: Here's a C version of the function, followed by the assembly code that implements it:

```
int max(int v1, int v2) {  
    if (v1 > v2)  
        return v1;  
    else  
        return v2;  
}
```

```
max          PROC                                ; Start of subroutine  
    push     ebp                                ; Save ebp  
    mov      ebp, esp                            ; Copy ebp to esp  
                                                    ; No local variables  
  
; CODE FOR: if (v1 > v2)  
    mov      eax, DWORD PTR 8[ebp]              ; eax = v1  
    cmp      eax, DWORD PTR 12[ebp]             ; Compare v1 to v2  
    jle      L1                                  ; Jump to L1 if v1 <= v2  
                                                    ; ((v1 > v2) is false)  
  
; CODE FOR: return v1;  
    jmp      L2                                  ; Jump to L2  
                                                    ; Return value (v1) is  
                                                    ; already in eax  
                                                    ; L2 is start of  
                                                    ; "cleanup" code  
  
; CODE FOR: else  
;           return v2;  
L1:          mov      eax, DWORD PTR 12[ebp]      ; Copy v2 into eax  
                                                    ; eax always holds  
                                                    ; function return value  
  
; CLEANUP  
L2:          pop      ebp                        ; Restore ebp  
    ret                                     ; Return from subroutine  
max          ENDP
```

c. `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.

Solution: Here's a C version of the function, followed by the assembly code that implements it:

```
void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

swap      PROC                                ; Start of subroutine
    push   ebp                                ; Save ebp
    mov    ebp, esp                            ; Copy ebp to esp
    sub    esp, 4                             ; Create space for temp

; CODE FOR: temp = *a
    mov    eax, DWORD PTR 8[ebp]              ; eax = address that "a"
                                                ; points to
    mov    ecx, DWORD PTR [eax]              ; ecx = value that "a"
                                                ; points to = *a
    mov    DWORD PTR -4[ebp], ecx            ; temp = *a

; CODE FOR: *a = *b
    mov    ecx, DWORD PTR 12[ebp]            ; ecx = address that "b"
                                                ; points to
    mov    edx, DWORD PTR [ecx]              ; edx = value that "b"
                                                ; points to = *b
    mov    DWORD PTR [eax], edx              ; *a = *b
                                                ; eax still holds address
                                                ; "a" points to

; CODE FOR: *b = temp;
    mov    eax, DWORD PTR -4[ebp]            ; eax = temp
    mov    DWORD PTR [ecx], eax              ; *b = temp
                                                ; ecx still holds address
                                                ; "b" points to

; CLEANUP
    mov    esp, ebp                            ; Clear space for temp
    pop    ebp                                ; Restore ebp
    ret                                         ; Return from subroutine
swap      ENDP
```