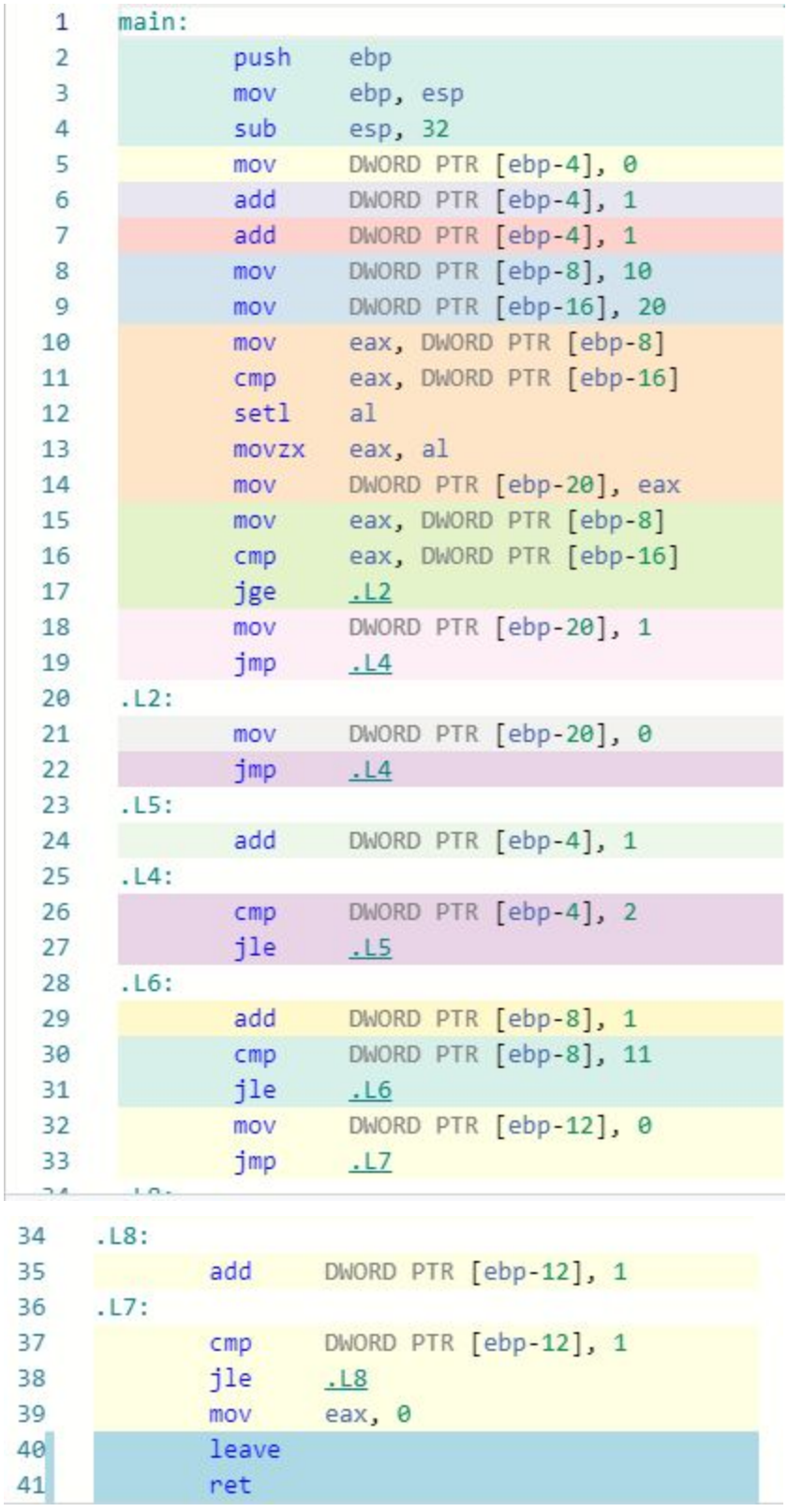
CSCE 541 Homework 2
Taylor Williamson
125007948
2/10/2020

1. In the subroutine, there are objects allocated on the stack on either side of the base pointer. Since arguments are pushed onto the stack and the base pointer is updated to mark the start of the subroutine memory space. Since the pointer values are decreasing as the stack grows, the values from [ebp+8], [ebp+12], and [ebp+16] are arguments for the subroutine while [ebp-4] and [ebp-8] are local variables of the subroutine.

2. Below is the C code and corresponding assembly code showing the differences between different C operations in assembly.

```
1   // Type your code here, or load an example.
2   int main(){
3       int i = 0;
4       i++;
5       i = i +1;
6
7       int x = 10, b = 20;
8       int c = (x<b) ? 1 : 0;
9
10      if(x<b){
11          c = 1;
12      }else{
13          c = 0;
14      }
15
16      while(i < 3){
17          i++;
18      }
19
20      do{
21          x++;
22      }while(x<12);
23
24      for(int j = 0; j < 2; j++){
25
26      }
27  }
```



```
1   main:
2           push    ebp
3           mov     ebp, esp
4           sub     esp, 32
5           mov     DWORD PTR [ebp-4], 0
6           add     DWORD PTR [ebp-4], 1
7           add     DWORD PTR [ebp-4], 1
8           mov     DWORD PTR [ebp-8], 10
9           mov     DWORD PTR [ebp-16], 20
10          mov     eax, DWORD PTR [ebp-8]
11          cmp     eax, DWORD PTR [ebp-16]
12          setl    al
13          movzx   eax, al
14          mov     DWORD PTR [ebp-20], eax
15          mov     eax, DWORD PTR [ebp-8]
16          cmp     eax, DWORD PTR [ebp-16]
17          jge     .L2
18          mov     DWORD PTR [ebp-20], 1
19          jmp     .L4
20  .L2:
21          mov     DWORD PTR [ebp-20], 0
22          jmp     .L4
23  .L5:
24          add     DWORD PTR [ebp-4], 1
25  .L4:
26          cmp     DWORD PTR [ebp-4], 2
27          jle     .L5
28  .L6:
29          add     DWORD PTR [ebp-8], 1
30          cmp     DWORD PTR [ebp-8], 11
31          jle     .L6
32          mov     DWORD PTR [ebp-12], 0
33          jmp     .L7
34  .L8:
35          add     DWORD PTR [ebp-12], 1
36  .L7:
37          cmp     DWORD PTR [ebp-12], 1
38          jle     .L8
39          mov     eax, 0
40          leave
41          ret
```

As we can see above, there is no difference between i++ and i = i+1, there was a significant efficiency increase when using ternary operators (orange) rather than if/else (green, pink, light

blue, and purple), and the loop method to use would be based mostly on its functionality to the program. Therefore, outside of the if/else and the drastically different control flow from the ternary operator, there is no significant change to the constructs at the base of these C operations.

3. In AT&T syntax, 8(%ebp) is equivalent to [ebp+8]. Therefore, this subroutine puts [ebp+8] into ecx, then [ebp+12] into eax. Now, the values in those registers are subtracted, [ebp+12] - [ebp+8], and the result is stored in ecx. This is then put back in eax to be returned. Therefore, the value that will be returned is [ebp+12]-[ebp+8].
4. Function 5 and 6 both have 3 arguments and they are the same. The values from 6 are pushed on the stack to be used as arguments for function 5.
5. Arguments are pushed onto the stack, function 5 is called, the result is returned, function 6 modifies the value of the stack pointer and then terminates the process.
6.

Assuming 8 bits

a	b	ZF	OF	CF	PF	AF	SF
1	1	1	0	0	Even - 1 Odd - 0	0	0
1	3	0	0	1	Even - 0 Odd - 1	0	1
100	-100	0	1	0	Even - 0 Odd - 1	0	1
1	2	0	0	1	Even - 1 Odd - 0	0	1

7. The value of the eax register at the end of the main function is 123434, since the starting address of the array is sent as input to the subroutine, which takes it and adds it to a running sum. There is a test which will always result in a non-jump condition, causing the code in .L3 to never execute. The value is placed into a word pointer and then returned, where the main function takes it and converts it back to a double word pointer and then returns the final value.
8. The return value of the program is 19.