CMSC 449
Malware Analysis

Recognizing C Code Constructs in Assembly

C Code Constructs

- **Code construct** – Code abstraction level that defines a particular operation, without specifying how it works
  - Loops, if statements, function calls, arrays, etc.
  - Or, if you like, how do C control structures and data structures get "rendered" at the assembly level?

- Important to recognize what these look like in assembly!

Accessing Arguments and Local Variables

- Function arguments are stored at offsets above EBP
  - Starting from 0x8, and incrementing by 0x4
  - First argument is [EBP+0x8], second argument is [EBP+0xC], …

- Local variables are stored at offsets below EBP
  - Offset depends on size of local variable, other variables
  - Example is [EBP-0x4]

Local and Global Variables

- Local variables can only be accessed from within the function in which they are defined
- In C, local variables include static, automatic, and register storage classes

- Global variables can be accessed from anywhere in a program
- In C, global variables are those in the extern storage class

Global Variable Example - C

```
int x = 1;
int y = 2;

void main() {
    x = x + y;
    printf("Total = %d\n", x);
}
```

Global Variable Example - Assembly

```
MOV     EAX, dword_40CF60       ; load x into EAX
ADD     EAX, dword_40C000       ; add y into EAX
MOV     dword_40CF60, EAX       ; store sum in x
MOV     ECX dword_40CF60        ; what's next?
PUSH    ECX
PUSH    offset aTotalD    ;"total = %d\n"
CALL    printf
```

Local Variable Example - C

```
void main(){
    int x = 1;
    int y = 2;
    x = x + y;
    printf("Total = %d\n", x);
}
```

Local Variable Example - Assembly

```
MOV     dword ptr [EBP-4], 0
MOV     dword ptr [EBP-8], 1
MOV     EAX, [EBP-4]
ADD     EAX, [EBP-8]
MOV     [EBP-4], eax
MOV     ECX, [EBP-4]
PUSH    ECX
PUSH    offset aTotalD    ; "total = %d\n"
CALL    printf
```

Identifying Arrays in Assembly

```
int i = 9;
int arr[64];
arr[i] = 50;
```

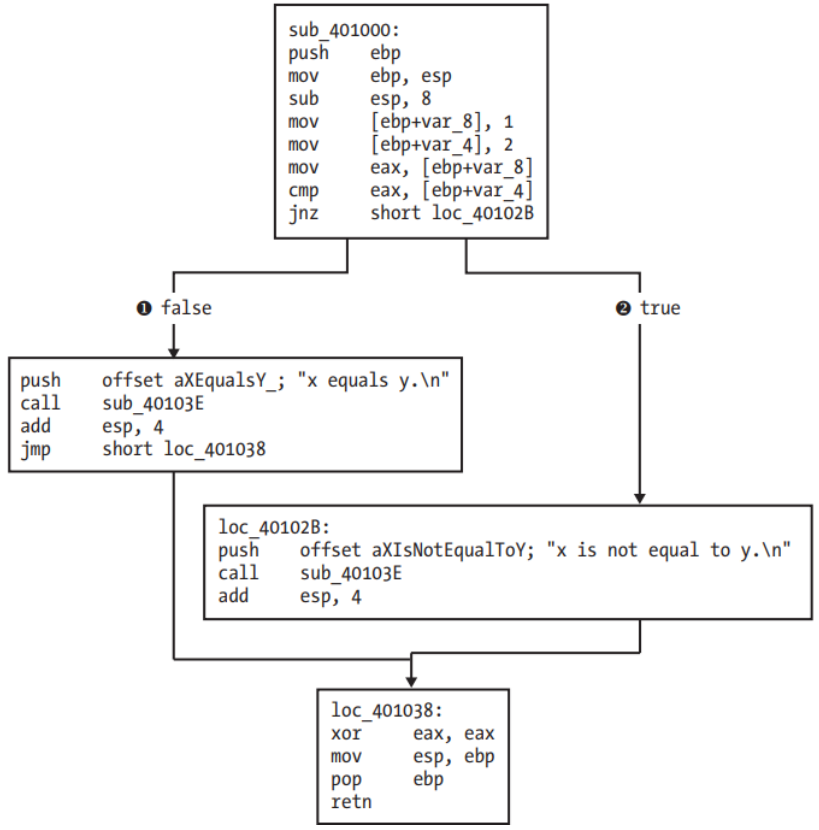
```
SUB    ESP, 0x44              ; Reserve 68 bytes on stack
MOV    [EBP-0x4], 0x9
MOV    EAX, [EBP-0x4]
MOV    EBX, 0x32
MOV    [EBP-0x44+EAX*4], EBX
```

If / Else Statement in Assembly

```
    CMP     EAX, EBX
    JNZ     short loc_40102B
    PUSH    offset aEAXeq_     ; "EAX == EBX\n"
    CALL    printf
    JMP     short loc_401038
loc_40102B:
    PUSH    offset aEAXneq_    ; "EAX != EBX\n"
    call printf
loc_401038:
```

Side Note- Disassembly Graphs

- Tools such as IDA Pro and Ghidra help visualize branching and loops
- This is another example of an if/else statement in assembly
- Much easier to understand how the control flow works!

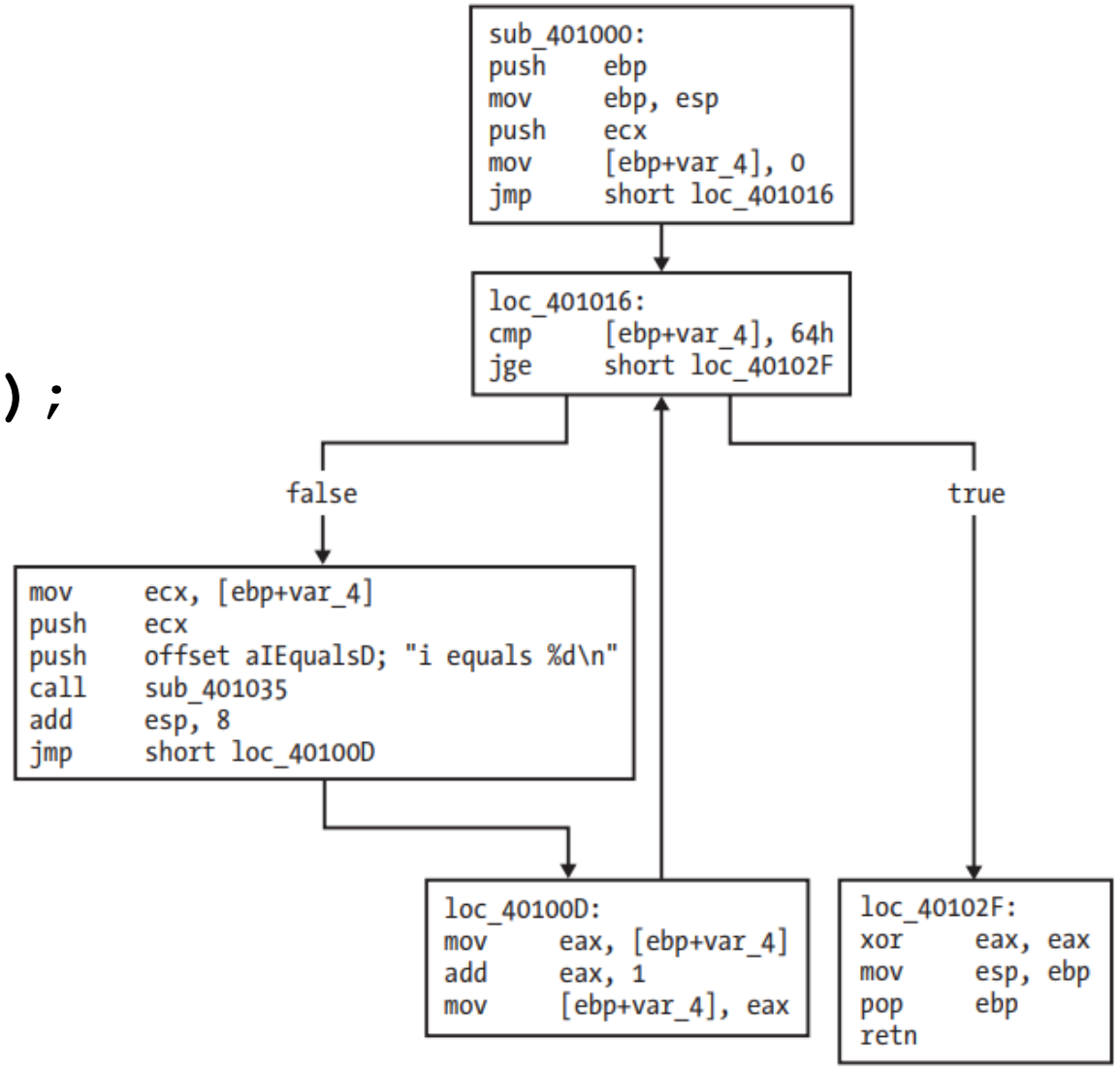


Recognizing While Loops in Assembly

```
    MOV    EAX, -1
loc_401044:
    CMP    EAX, 0
    JNZ    short loc_401063
    CALL   myFunc
    JMP    short loc_401044
loc_401063
    ...
    RET
```

Recognizing For Loops in Assembly

```
void loop_func(){
    int i;
    for(i = 0; i < 100; i++){
        printf("i equals %d\n");
    }
    return;
}
```



If-Style Switch Statements

```
switch(i){
    case 0:
        printf("i = %d", i);
        break;
    case 1:
        printf("i = %d", i);
        break;
    default:
        break;
}
```

If-Style Switch Statements

```
        CMP     [EBP-0x4], 0
        JZ      short loc_401027
        CMP     [EBP-0x4], 1
        JZ      short loc_40103D
        jmp short loc_401067      ; Default
loc_401027
        ...
loc_40103D
        ...
```

Jump Table Switch Statements

```
    MOV   EDX, [EBP-0x4]
    JMP   ds:off_401088[EDX*4]
loc_40106E ; case 0
    ...
loc_401042 ; case 1
    ...
loc_401058 ; case 2
    ...
```

401088	00	40	10	6E
40108C	00	40	10	42
401090	00	40	10	58

Jump Tables in Disassembly Graphs