EE 213 Computer Organization and Assembly Language

Week # 2, Lecture # 4

22nd Dhu'l-Hijjah, 1439 A.H

3rd September 2018

These slides contains materials taken from various sources. I fully acknowledge all copyrights.

Minds open...

... Laptops closed

This presentation helps in delivering the lecture.

Take notes, interact and read text book to learn and gain knowledge.

Revision of Topics from Previous Lecture

- Cache
- Memory address range
- Hex to Binary
- Binary to Hex

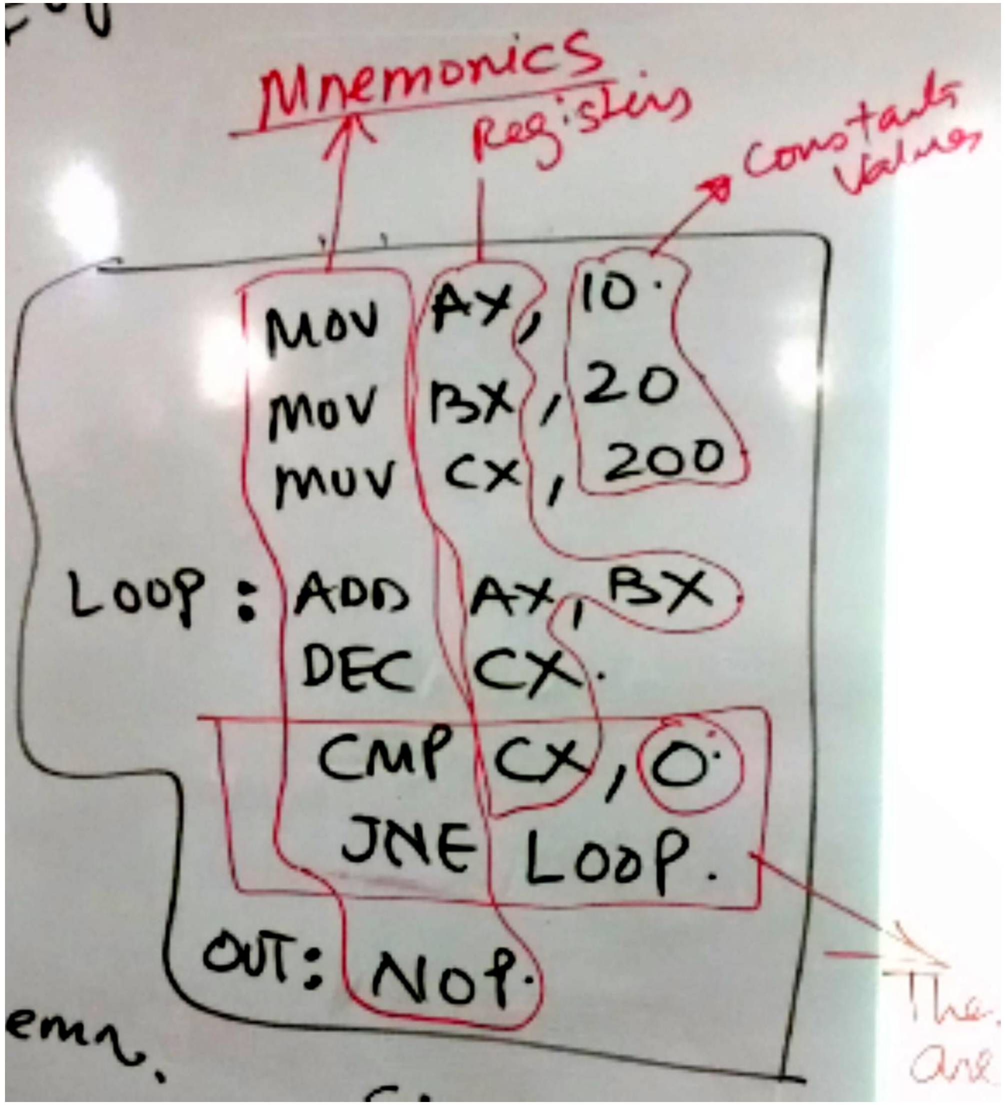
Learn in Lab. All labs contents are part of theory syllabus.

- Instruction Fetch and Execute
- Assembly Programs
  - High-level language are human friendly doesn’t shows hardware related details. Executable code contains one and zero which are difficult for humans.
  - Need a way to write programs that show processor details.
  - Assembly Language fills this gap by providing language statements which are closer to micro-architecture elements.
  - Therefore, the key goal of learning assembly is to understanding how HLL are executed on micro-architecture for better computational thinking.

Today’s Topics

- Understanding take-home assembly code
    - See whiteboard snaps for topics covered.
- High-Level code <-> Assembly code <-> Machine code
- What is machine code?

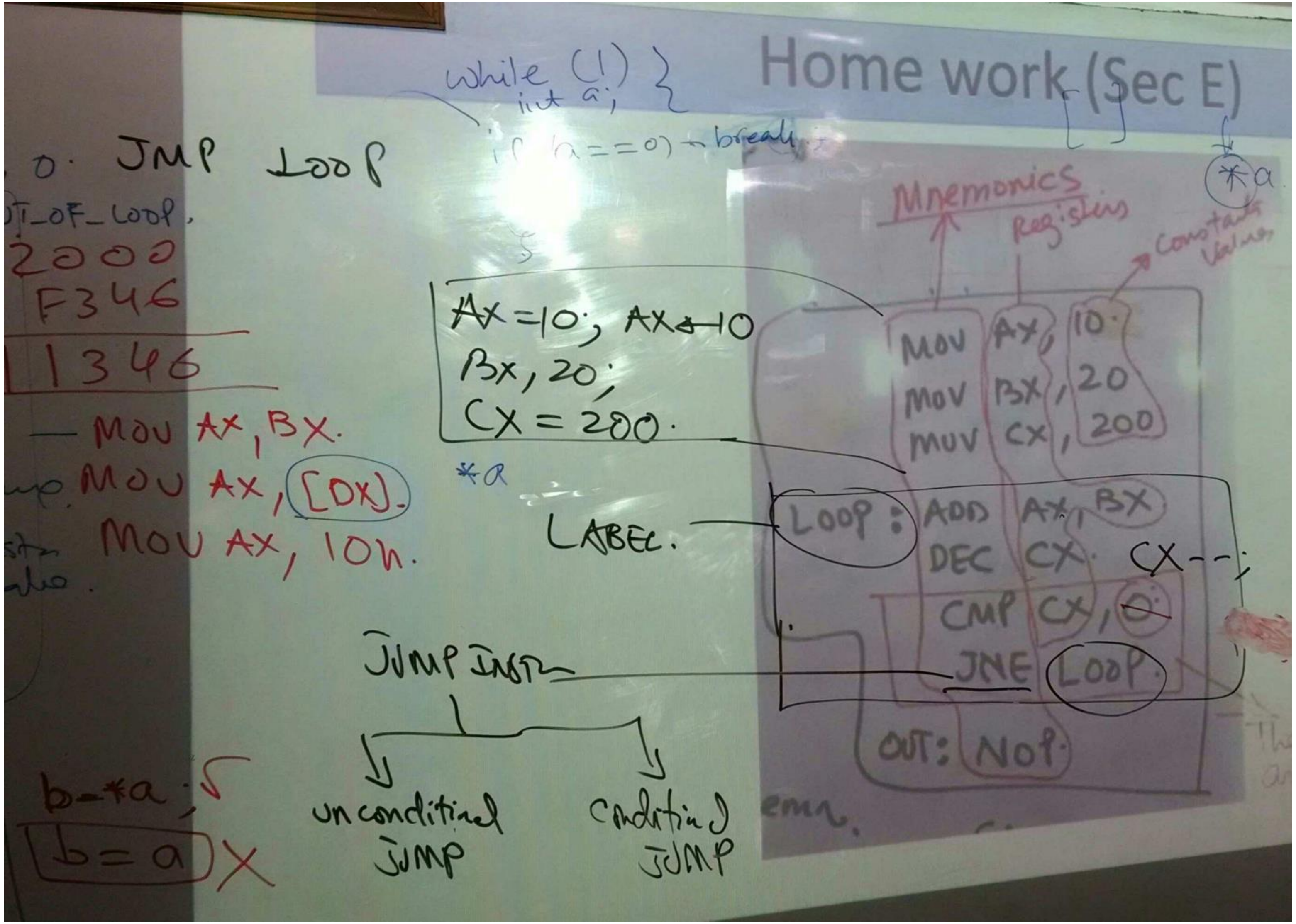
Homework Assembly code



```
      MOV AX, 10.
      MOV BX, 20
      MUV CX, 200

LOOP: ADD AX, BX
      DEC CX.
      CMP CX, 0.
      JNE LOOP.

OUT:  NOP.
```

Mnemonics
Registers
Constants Values
The
are
emn.

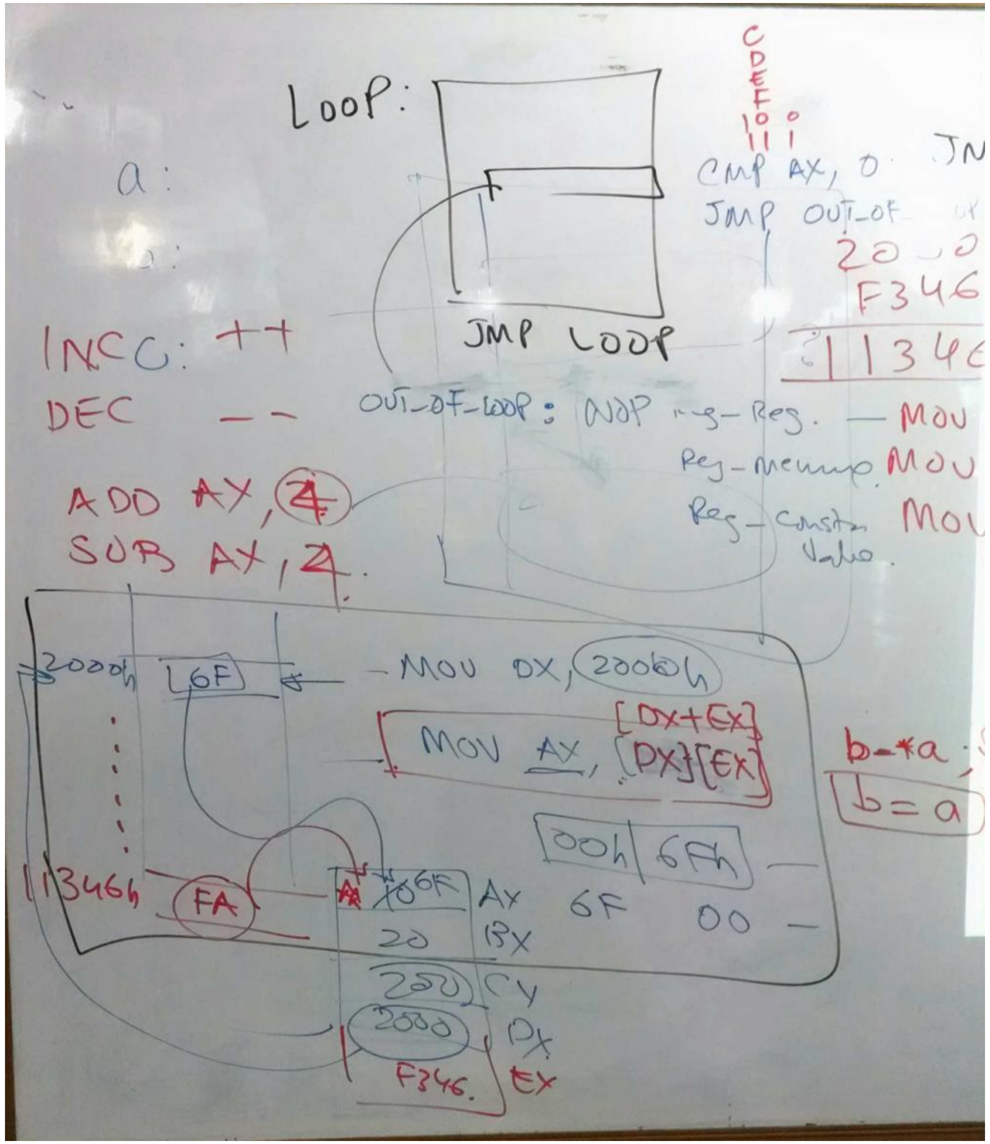
Explanation of Homework Assembly code



```
MOV AX, 10
MOV BX, 20
MUV CX, 200
Loop: ADD AX, BX
DEC CX
CMP CX, 0
JNE LOOP.
OUT: NOP.
```

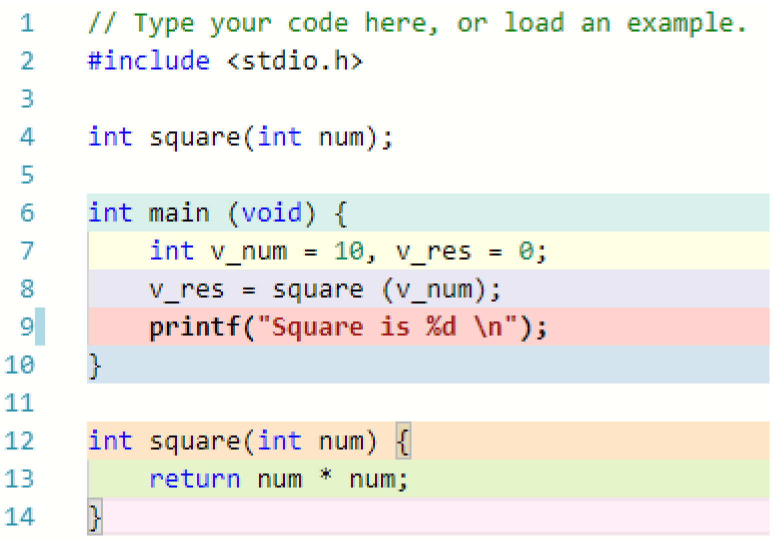
- Unconditional Jump (JMP)
- Conditional Jump (using two instructions CMP and JNE)
- Operands are in processor register or memory (RAM)
- Memory Operands are accessed either directly specifying memory address or loading address in register and use it as a pointer.
- Hex addition for address calculations
- Labels for loops and jumps just like GOTO Label in BASIC Language
- NOP – No Operation Instructions

Explanation of Homework Assembly code

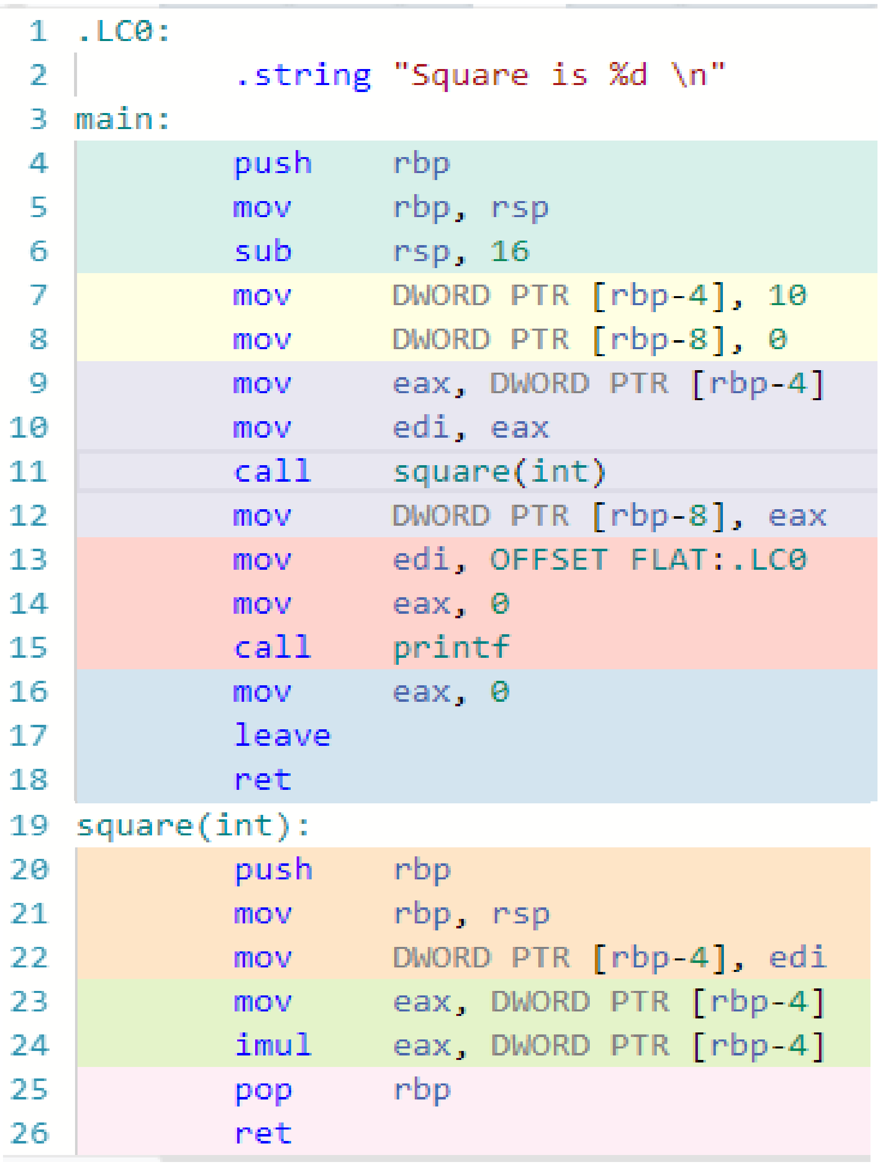


- INC AX means increment processor register value by one
- DEC AX means decrement processor register value by one
- ADD AX, 2 ; AX <- AX + 2
- SUB AX, 2; AX <- AX – 2
- The “2” in the above instructions is call a constant. It could be a character like ‘a’
- MOV DX, 2000h
  MOV AX, [DX]
- means move value stored at memory address 2000h into AX (two bytes will be moves because AX is 16 bits)

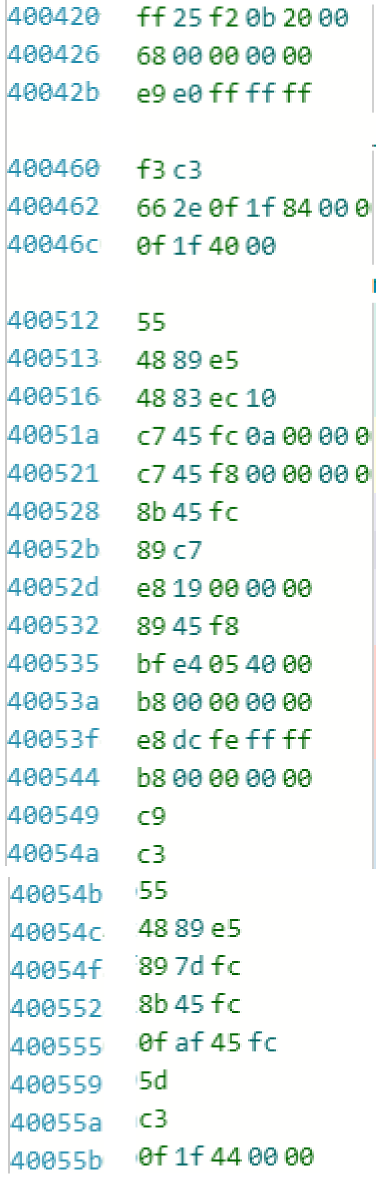
High-Level code <-> Assembly code <-> Machine code



```c
// Type your code here, or load an example.
#include <stdio.h>

int square(int num);

int main (void) {
    int v_num = 10, v_res = 0;
    v_res = square (v_num);
    printf("Square is %d \n");
}

int square(int num) {
    return num * num;
}
```



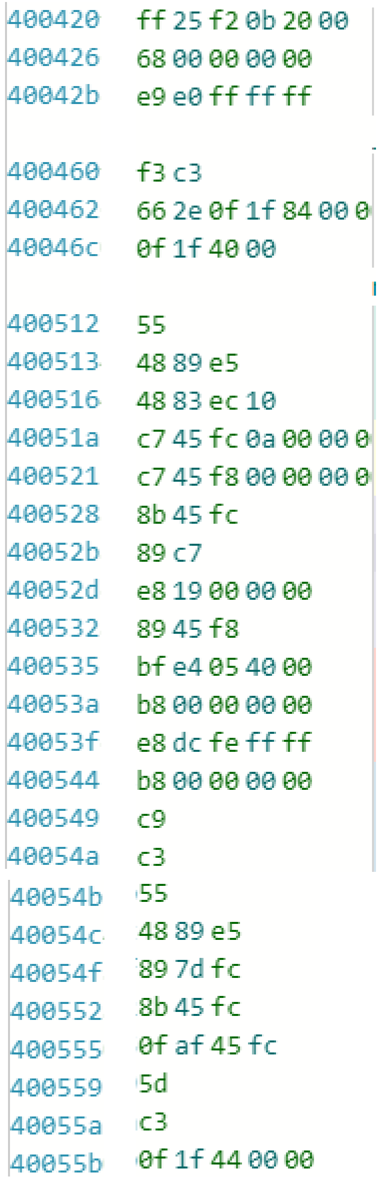
```asm
.LC0:
        .string "Square is %d \n"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 16
        mov     DWORD PTR [rbp-4], 10
        mov     DWORD PTR [rbp-8], 0
        mov     eax, DWORD PTR [rbp-4]
        mov     edi, eax
        call    square(int)
        mov     DWORD PTR [rbp-8], eax
        mov     edi, OFFSET FLAT:.LC0
        mov     eax, 0
        call    printf
        mov     eax, 0
        leave
        ret
square(int):
        push    rbp
        mov     rbp, rsp
        mov     DWORD PTR [rbp-4], edi
        mov     eax, DWORD PTR [rbp-4]
        imul    eax, DWORD PTR [rbp-4]
        pop     rbp
        ret
```



```
400420   ff 25 f2 0b 20 00
400426   68 00 00 00 00
40042b   e9 e0 ff ff ff

400460   f3 c3
400462   66 2e 0f 1f 84 00 0
40046c   0f 1f 40 00

400512   55
400513   48 89 e5
400516   48 83 ec 10
40051a   c7 45 fc 0a 00 00 0
400521   c7 45 f8 00 00 00 0
400528   8b 45 fc
40052b   89 c7
40052d   e8 19 00 00 00
400532   89 45 f8
400535   bf e4 05 40 00
40053a   b8 00 00 00 00
40053f   e8 dc fe ff ff
400544   b8 00 00 00 00
400549   c9
40054a   c3
40054b   55
40054c   48 89 e5
40054f   89 7d fc
400552   8b 45 fc
400555   0f af 45 fc
400559   5d
40055a   c3
40055b   0f 1f 44 00 00
```

What is machine code?

- Machine code is a computer program written in machine language instructions that can be executed directly by a processor.
- Machine code is strictly numerical and may be regarded as the lowest-level representation of a program or as a hardware-dependent programming language.
- It is possible to write programs directly in machine code, but it is tedious and error prone to manage individual bits and calculate numerical addresses and constants manually.
- Programs are very rarely written directly in machine code in modern contexts. Machine coding is done for low level debugging, program patching, etc.



```
400420   ff 25 f2 0b 20 00
400426   68 00 00 00 00
40042b   e9 e0 ff ff ff

400460   f3 c3
400462   66 2e 0f 1f 84 00 0
40046c   0f 1f 40 00

400512   55
400513   48 89 e5
400516   48 83 ec 10
40051a   c7 45 fc 0a 00 00 0
400521   c7 45 f8 00 00 00 0
400528   8b 45 fc
40052b   89 c7
40052d   e8 19 00 00 00
400532   89 45 f8
400535   bf e4 05 40 00
40053a   b8 00 00 00 00
40053f   e8 dc fe ff ff
400544   b8 00 00 00 00
400549   c9
40054a   c3
40054b   55
40054c   48 89 e5
40054f   89 7d fc
400552   8b 45 fc
400555   0f af 45 fc
400559   5d
40055a   c3
40055b   0f 1f 44 00 00
```