Assembly

- ESP contains address of stack first byte beyond the top
- On Quiz 3, 12b is 0018ff8C
  12c is 0018FE8C
- push - 1.) subtracts from ESP 2.) copies bytes to stack
  pop - 1) copies data from stack 2.) pointer moved
- Quiz on Thursday, 9.15

- Integer constants
  decimal - add ax 24; add ax 24d
  binary - add ax 100b,
  hex - add ax 64h,
  character - add ax 'A'
  string - add ax 'John went'; add ax "John went"
- Integer expressions
  () - parenthesis
  * ÷ / - multiply & divide
  mod - modulus
  + - - add, subtract
- Reserved words
  instruction mnemonics - add, sub, mul, mov
  registers - eax, ebx...
  directives - tells MASM what to do
  attributes - messages to assembly - byte, dword
  operators - +, -, ()
  pre-defined symbols - @
- Identifiers
  must begin with A-Z, a-z, _, $, @, ?
  can be up to 247 characters
  normally not case sensitive, unless switch is turned on
  must start A-Z, a-z, other characters can be numerical

Assembly

- Directives
  not executable
  define variable or procedures
  can assign names to memory segments or code
  case insensitive
  Example - .486, .model
  first three lines of program should be
  .486
  .model flat
  .stack 100h

- .asm → assembler → .lst
  .obj → linker
  .obj
  .lib

- Instructions - 4 parts
  [label] operation [operands] [comment]
  label - assigned address of the instruction that follows, label appears in the symbol table
- to debug, _start:
  first line under .code
- last thing is PUBLIC start
- last line is END
  operation - mov
- b for byte, w for word, d for dword, q for qword, str for string (use these for variable names)
- CRLF - 10 13, 0
- strPrompt1 byte "Enter first number", 0
- to use Bailey's macro, follow string with a 0
- receiving operand is always first
-

Assembly

- each instruction must have a comment!
- add ebx, 4 ; ebx points to next integer value

```
; Program
; Programer
; Course
```

Write why? for comments

- Assembly cannot do memory to memory operations, must use registers!
- add wX, wY - WRONG!
- mov ax, wX ; wZ =
  add ax, wY ; wX +
  mov wZ, ax ; wY
- sub op1, op2 ; op1 gets result of op1 - op2
- Nop
  mov eax, dVal ; dVal : : 27d
  loader always makes sure it is on a double word boundary
  nop is used to align, takes up one byte of storage,
  speeds up processing

- Quiz 4 will be over this review

```
; Program        filename.asm          line up!
; Programer      Name
; Course         CSCI 2160-201
; Assignment
; Date
; Purpose
```

```
.486
.model flat
.stack 100h
```

(review handout on D2L)

- display prompt 1
  get value 1
  display prompt 2
  get value 2
  add values
  convert answer
  display answer

- .code

Quiz 4 BWB.asm

```
_start:
    puts    strPrompt1          ; display first input prompt
    gets    strInput, 10        ; get first value from keyboard
    stringToNum strInput, dXval ; convert to a true binary value
    puts    strPrompt2          ; display 2nd input prompt
    gets    strInput, 10        ; get second value fm keyboard
    stringToNum, dYval          ; convert to a true binary value
    mov     eax, dXval          ;   dXVal =
    add     eax, dYval          ;     dYval +
    mov     dXVal, eax          ;       dXval
    numToString strOut, dXval   ; convert the binary value to a numeric string
    puts    strOut              ; display string

    INVOKE ExitProcess, 0
    PUBLIC _start
    END
```