More Pentium Instructions

October 20

CSC201 Section 002

Fall, 2000

Branching

- For the Pentium, there are both signed and unsigned branch instructions
  - we will ignore (and the book ignores) unsigned branch instructions

Jump (Branch) Instructions

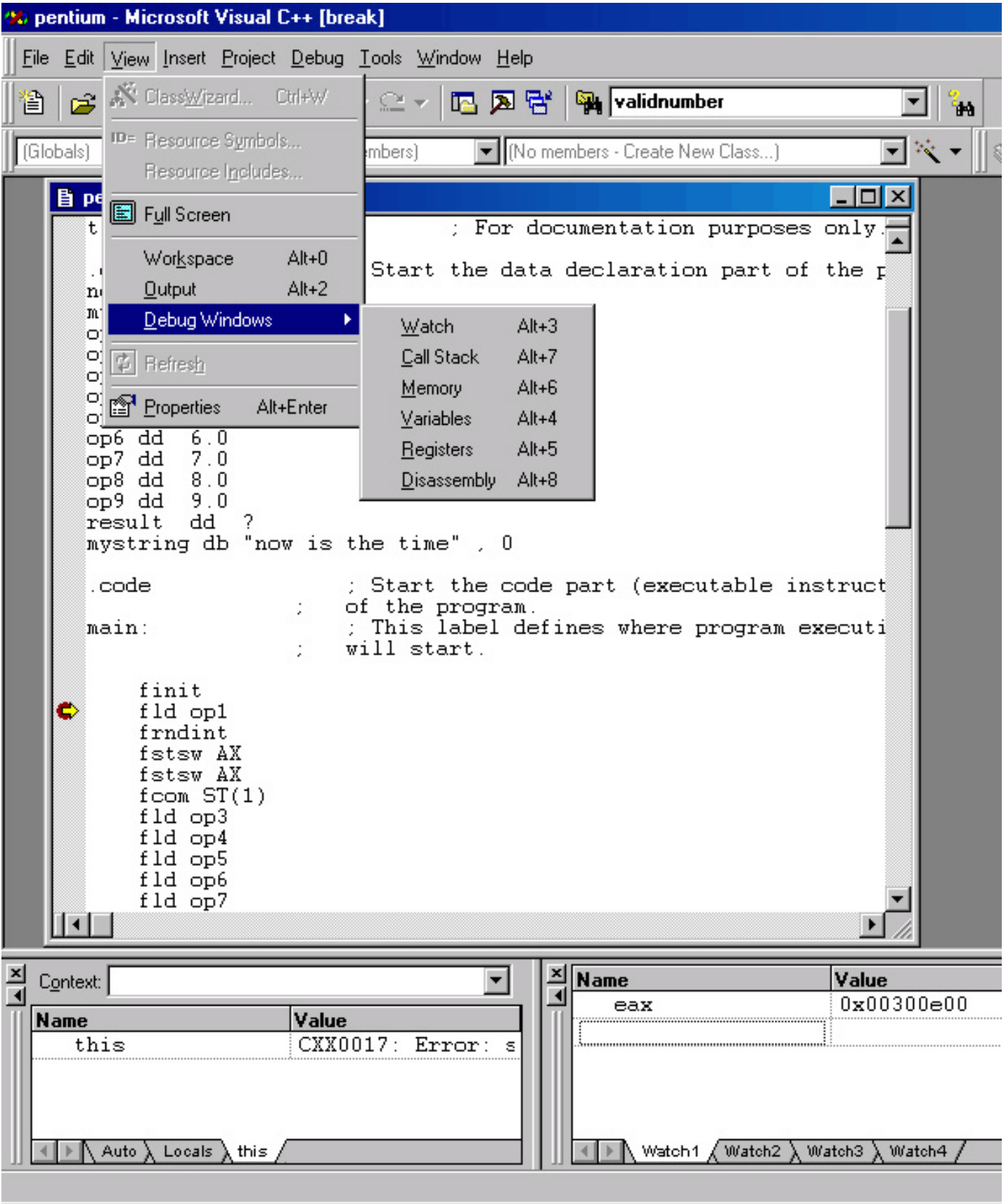
SASM	Pentium	Condition
br	jmp	(unconditional)
bg	jg	If greater than
bgez	jge	If greater than or equal
bl	jl	If less than
blez	jle	If less than or equal
bez	je (or jz)	If equal (or if ZF = 1)
bnz	jne (or jnz)	If not equal (or if ZF = 0
	jo, jno	If OF = 1, 0
	jc, jnc	If CF = 1, 0
	js, jns	If SF = 1, 0

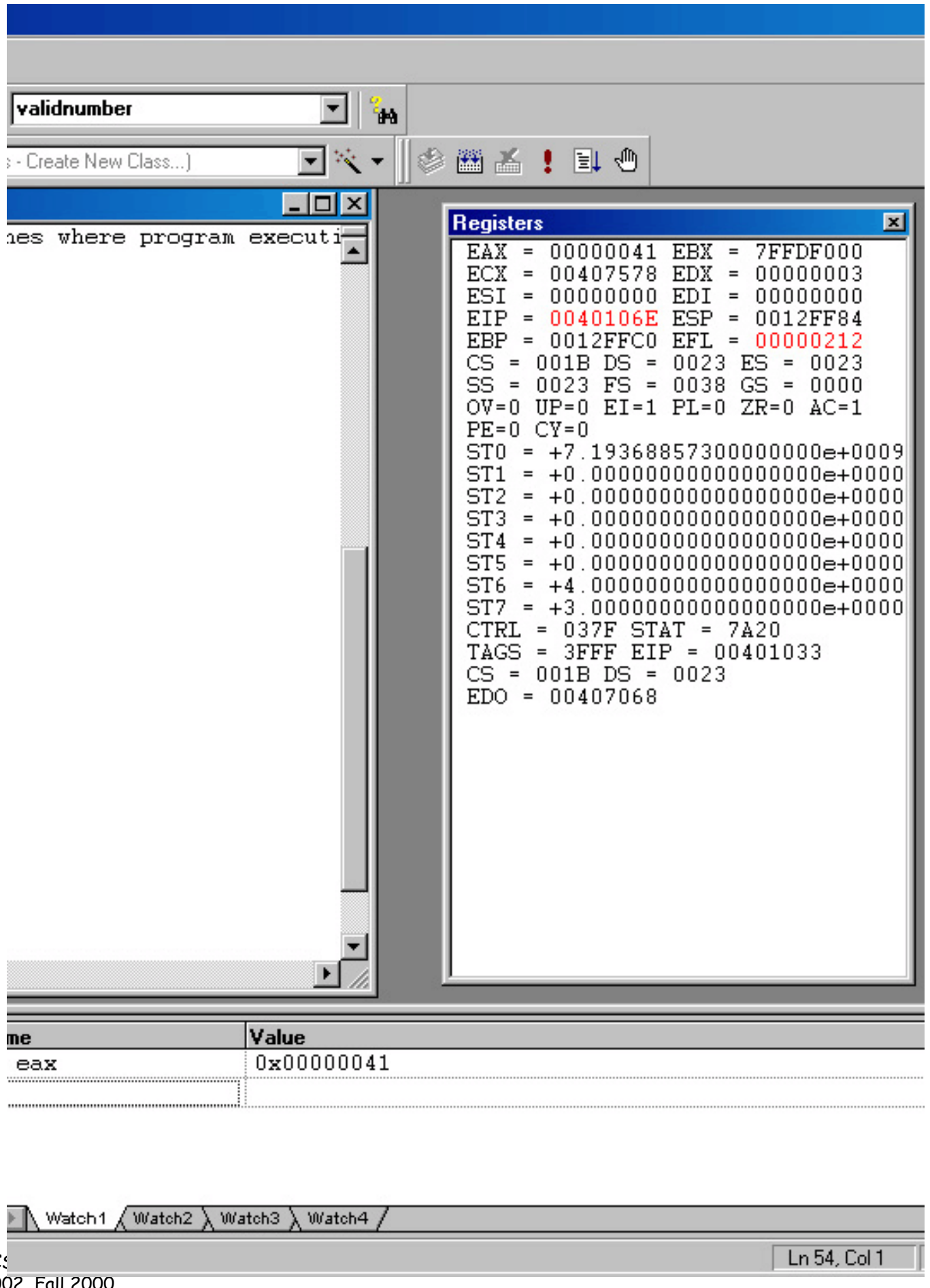
I/O Macros

- There are no "real" I/O instructions
  - only calls to the operating system for I/O services
- I/O capabilities provided by Windows + pmacros.inc
  - "put_ch reg/mem" = output least significant byte of operand
  - "get_ch" = input byte into AL register
  - "put_str mem" = output null-terminated string starting at location mem
- No formatting or automatic conversions!

Pentium Debugging

- Use the "View → Debugging → Registers" command in Visual Studio to see the register values
- Use the "Watch" window to change the register values




validnumber
- Create New Class...)
nes where program executi
Registers
EAX = 00000041 EBX = 7FFDF000
ECX = 00407578 EDX = 00000003
ESI = 00000000 EDI = 00000000
EIP = 0040106E ESP = 0012FF84
EBP = 0012FFC0 EFL = 00000212
CS = 001B DS = 0023 ES = 0023
SS = 0023 FS = 0038 GS = 0000
OV=0 UP=0 EI=1 PL=0 ZR=0 AC=1
PE=0 CY=0
ST0 = +7.19368857300000000e+0009
ST1 = +0.00000000000000000e+0000
ST2 = +0.00000000000000000e+0000
ST3 = +0.00000000000000000e+0000
ST4 = +0.00000000000000000e+0000
ST5 = +0.00000000000000000e+0000
ST6 = +4.00000000000000000e+0000
ST7 = +3.00000000000000000e+0000
CTRL = 037F STAT = 7A20
TAGS = 3FFF EIP = 00401033
CS = 001B DS = 0023
EDO = 00407068
me
Value
eax
0x00000041
Watch1
Watch2
Watch3
Watch4
Ln 54, Col 1

Floating Point

- Warning: I'm omitting a *lot* of details!
  - including a lot of legal instruction formats
- The floating point unit is logically separate from the rest of the CPU
  - first implemented as a coprocessor
- FPU Status Word
  - bit 14 = C3, bit 10 = C2, bit 8 = C0

Pentium Floating Point Registers

- 8 80-bit registers (double-precision)
- Organized (more or less) as a stack
  - numbered ST(0) (or ST, top of stack) to ST(7) (bottom of stack)

Moving Data To/From FPU

- fld: push value onto top of FP stack
  - operand = mem32 / mem64 / ST(i) (i calculated before pushing)
  - also, "fldz" means push zero onto stack
- fstp: pop value from top of FP stack
  - operand = mem32 / mem64 / ST(i) (i calculated before pushing)

FP Add/Subtract

- faddp: pop top two values, add, push result back onto stack
  - operands = ST(1), ST
- fsubp: pop top two values, subtract ST from ST(1), push result back onto stack
  - operands = ST(1), ST
- fmulp: pop top two values, multiply, push result back onto stack
  - operands = ST(1), ST
- fdivp: pop top two values, divide ST(i) by ST, push result back onto stack
  - operands = ST(1), ST

Other FPU Instructions

- fchs: change sign of top of stack
  - no operands
- frndint: round value of top of stack to nearest integer
  - no operands
  - fairly easy to convert result into integer representation

Flow of Control

- fcom: compare ST with operand and set C3--C0
  - operands: mem32 / mem64 / ST(i)
- Flag interpretations
  - ST = operand: C3 = 1, C2 = C0 = 0
  - ST > operand: C3 = C2 = C0 = 0
  - ST < operand: C0 = 1, C3 = C2 = 0
- fstsw: store status word into AX register
  - operand: AX required (why?)
  - Then can do bit testing on AX register and jump based on conditions

Stack-Based Expression Evaluation

Example: v = w + x * y - z

Instruction	Top of stack ST	ST(1)	ST(2)
finit	-	-	-
fld w	w	-	-
fld x	x	w	-
fld y	y	x	w
fmulp ST(1), ST	X*y	w	
faddp ST(1), ST	W+(x*y)	-	-
fld z	z	W+(x*y)	-
fsubp ST(1), ST	W+(x*y)-z	-	-
fstp v	-	-	-

More Expression Evaluation

- Another example

a = b * (c – d) / (e – b) + 10

```
fld   b
fld   c
fld   d
fsubp ST(1), ST
fmultp ST(1), ST
fld   e
fld   b
fsubp ST(1), ST
fdivp ST(1), ST
fld   10
faddp ST(1), ST
fstp a
```