MathWorks Compiler Course – Day 1

- Who is Bill McKeeman?
  - 10th year at MathWorks, started JIT project

MathWorks Compiler Course – Day 1

- Who is Bill McKeeman?
  - 10th year at MathWorks, started JIT project
  - 50th year of teaching, USNA, Stanford, UCSC, Toronto, Wang, Harvard, Dartmouth

MathWorks Compiler Course – Day 1

- Who is Bill McKeeman?
- What is xcom?
  - Teaching compiler implemented in C++ used at Dartmouth.

MathWorks Compiler Course – Day 1

- Who is Bill McKeeman?
- What is xcom?
  - Teaching compiler implemented in C++ used at Dartmouth.
  - Also implemented in M on MathWorks File Exchange

MathWorks Compiler Course – Day 1

- Who is Bill McKeeman?
- What is xcom?
  - Teaching compiler implemented in C++ used at Dartmouth.
  - Also implemented in M on MathWorks File Exchange
  - Implements a language called X

MathWorks Compiler Course – Day 1

- What is xcom?
  - Teaching compiler implemented in C++ used at Dartmouth.
  - Also implemented in M on MathWorks File Exchange
  - Implements a language called X
  - The MATLAB JIT came from a C version of xcom

MathWorks Compiler Course – Day 1

- What is xcom?
- What will be learned?
  - How to turn X source into Intel x86 code, and execute it.

MathWorks Compiler Course – Day 1

- What is xcom?
- What will be learned?
  - How to turn X source into Intel x86 code, and execute it.
  - About 6 deep results, some mathematical notation

MathWorks Compiler Course – Day 1

- What is xcom?
- What will be learned?
  - How to turn X source into Intel x86 code, and execute it.
  - About 6 deep results, some mathematical notation
  - Lots of coding tricks

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer
  - Operator := forces x to integer

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer
  - Operator := forces x to integer
  - Value of y is undefined, therefore input

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer
  - Operator := forces x to integer
  - Value of y is undefined, therefore input
  - Value of x is unused, therefore output

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer
  - Operator := forces x to integer
  - Value of y is undefined, therefore input
  - Value of x is unused, therefore output
  - (try it)

MathWorks Compiler Course – Day 1

- >> xcom x:=y+1
  - Operator + forces y to integer
  - Operator := forces x to integer
  - Value of y is undefined, therefore input
  - Value of x is unused, therefore output
  - (try it)
- Where the course is heading:

  >> xcom -asmDump -noExecute x:=y+1

  x86 code dump, 32 bytes code=0xbdb3e00

  5589e58b75088b8604000000b90100000001c889860000000b800000000c9c3

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’
  - Type of x and y is undefined

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’
  - Type of x and y is undefined
- >> xcom ‘x,y := 1,2; x,y := y,x’

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’
  - Type of x and y is undefined
- >> xcom ‘x,y := 1,2; x,y := y,x’
  - No output

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’
  - Type of x and y is undefined
- >> xcom ‘x,y := 1,2; x,y := y,x’
  - No output
- >> xcom ‘x,y := 1,2; x,y := y,x; a,b := x,y’
  - Exchanges x and y

MathWorks Compiler Course – Day 1

- >> xcom ‘x,y := y,x’
  - Type of x and y is undefined
- >> xcom ‘x,y := 1,2; x,y := y,x’
  - Exchanges x and y; no output
- >> xcom ‘x,y := 1,2; x,y := y,x; a,b := x,y’
  - Exchanges x and y; displays result
  - (try it)

MathWorks Compiler Course – Day 1

` FILE: divmod.x

` PURPOSE: function to capture quotient and remainder

```
quotient := a/b;
remainder := a//b;
```

MathWorks Compiler Course – Day 1

` FILE: divmod.x

` PURPOSE: function to capture quotient and remainder

```
quotient := a/b;
remainder := a//b;
```

- >> xcom x/divmod.x
  - (try it)

MathWorks Compiler Course – Day 1

Look at mxcom\doc\ReferenceX.pdf

Look at sin.x (next two pages)

```
` FILE:    sin.x
` PURPOSE: approximate sin(x) with x - x^3/6 + x^5/120 - ...
` USAGE:   xcom x/sin.x    (from command line)
`          xout := sin := xin  (in X source)

  ` -------- normalize argument x -------------------
  if xin < 0.0 ? xa := -xin; :: xin >= 0.0 ? xa := xin; fi;

  pi2 := 2.0*3.141592653589793;
  cut := 7.0;
  if xa > cut ?      ` get x into reasonable range
     red := r2i(xa/pi2)-1;
     xa  := xa - i2r(red)*pi2;
  :: xa <= cut ?     ` do nothing
  fi;
  xi := xa;          ` x^i
  x2 := xi*xi;       ` x^2

  ` --------- setup iteration ----------------
  i   := 0.0;        ` i
  sum := 0.0;        ` running sum
  fac := 1.0;        ` i!
  del := 1.0;        ` get started
  sg  := 1.0;        ` sign
```

```
` ---------- sum series -------------------
do del > 0.0000001 ?
  del := xi/fac;     ` non-negative increment
  sum := sum + sg*del;
  sg  := -sg;        ` alternate sign
  i   := i+2.0;      ` step 2
  fac := fac*i*(i+1.0);
  xi  := xi*x2;      ` mul by x^2
od;

` ------ normalize result sin(-x) = - sin(x) ------------
if xin < 0.0 ? xout := -sum; :: xin >= 0.0 ? xout := sum; fi
```

MathWorks Compiler Course – Day 1

Debugging an X program

- try something simpler
- put in intermediate assignments
- look at –asmTrace output

```
do del > 0.0000001 ?
    del := xi/fac;      ` non-negative increment
delLast := del;
    sum := sum + sg*del;
    sg  := -sg;       ` alternate sign
    i   := i+2.0;     ` step 2
    fac := fac*i*(i+1.0);
    xi  := xi*x2;     ` mul by x^2
  od;
```