Py65: Microcontroller Simulation with Python

PyWorks 2008

Mike Naberezny

Maintainable Software

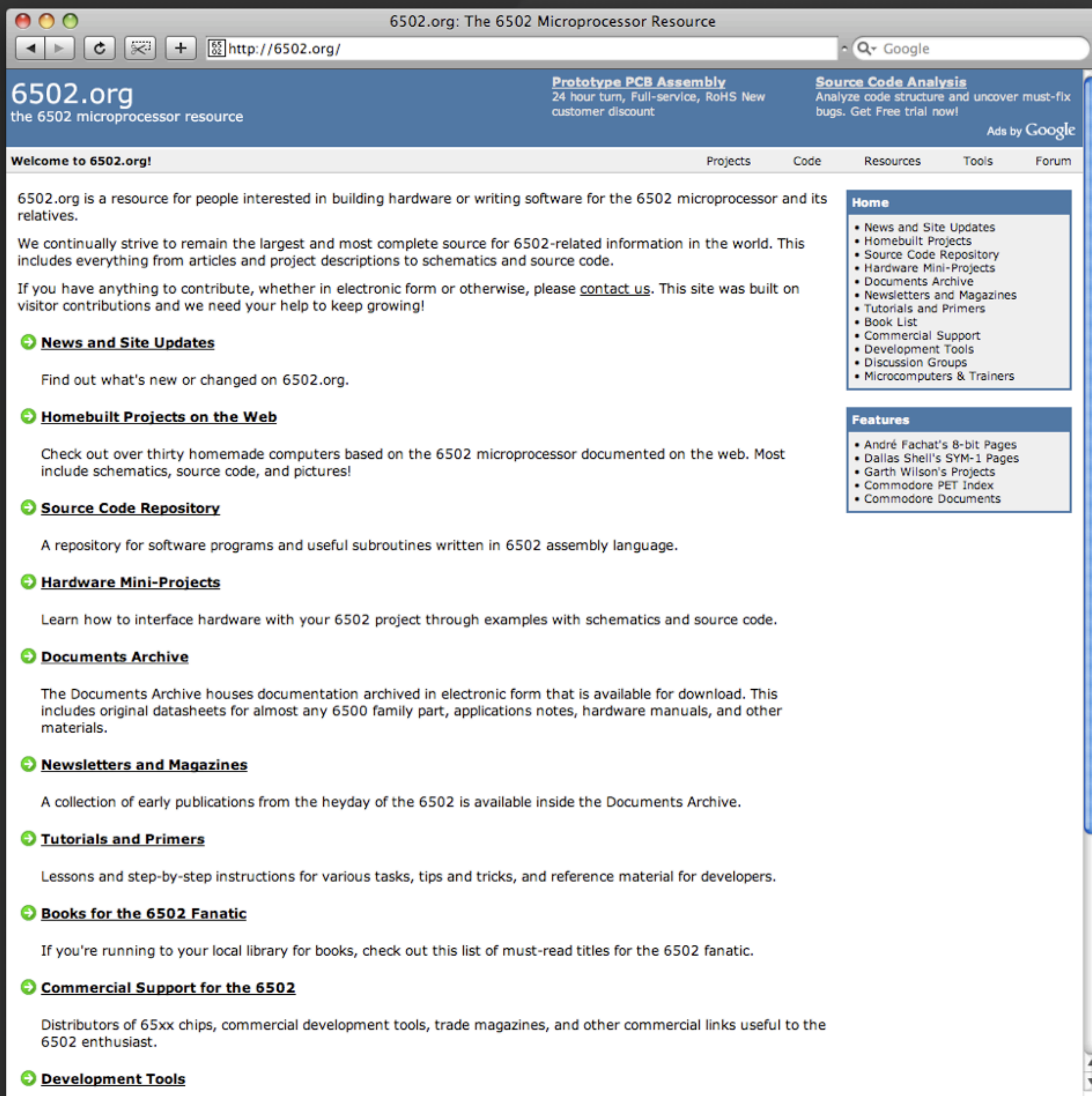
http://maintainable.com

About Me

- http://mikenaberezny.com
- http://maintainable.com
- http://ohloh.net/accounts/mnaberez

Maintainable Software

About Me



- http://6502.org
- Over 10 years online
- Gigabytes of 6502 technical information

Maintainable Software

About Py65

- Simulation of 65xx family hardware components as Python objects
- Very new project: started July 2008
- Target audience
  - Engineers working on embedded software
  - Students learning how computers work

Maintainable Software

About Py65

```
def test_inx_increments_x(self):
  mpu = MPU()
  mpu.x = 0x09
  mpu.memory[0x0000] = 0xE8 #=> INX
  mpu.step()
  self.assertEquals(0x0001, mpu.pc)
  self.assertEquals(0x0A, mpu.x)
  self.assertEquals(0, mpu.flags & mpu.ZERO)
  self.assertEquals(0, mpu.flags & mpu.NEGATIVE)
```

- Very low-level simulation
- Even this can be easier than dealing with the real hardware

Maintainable Software

Today’s Talk

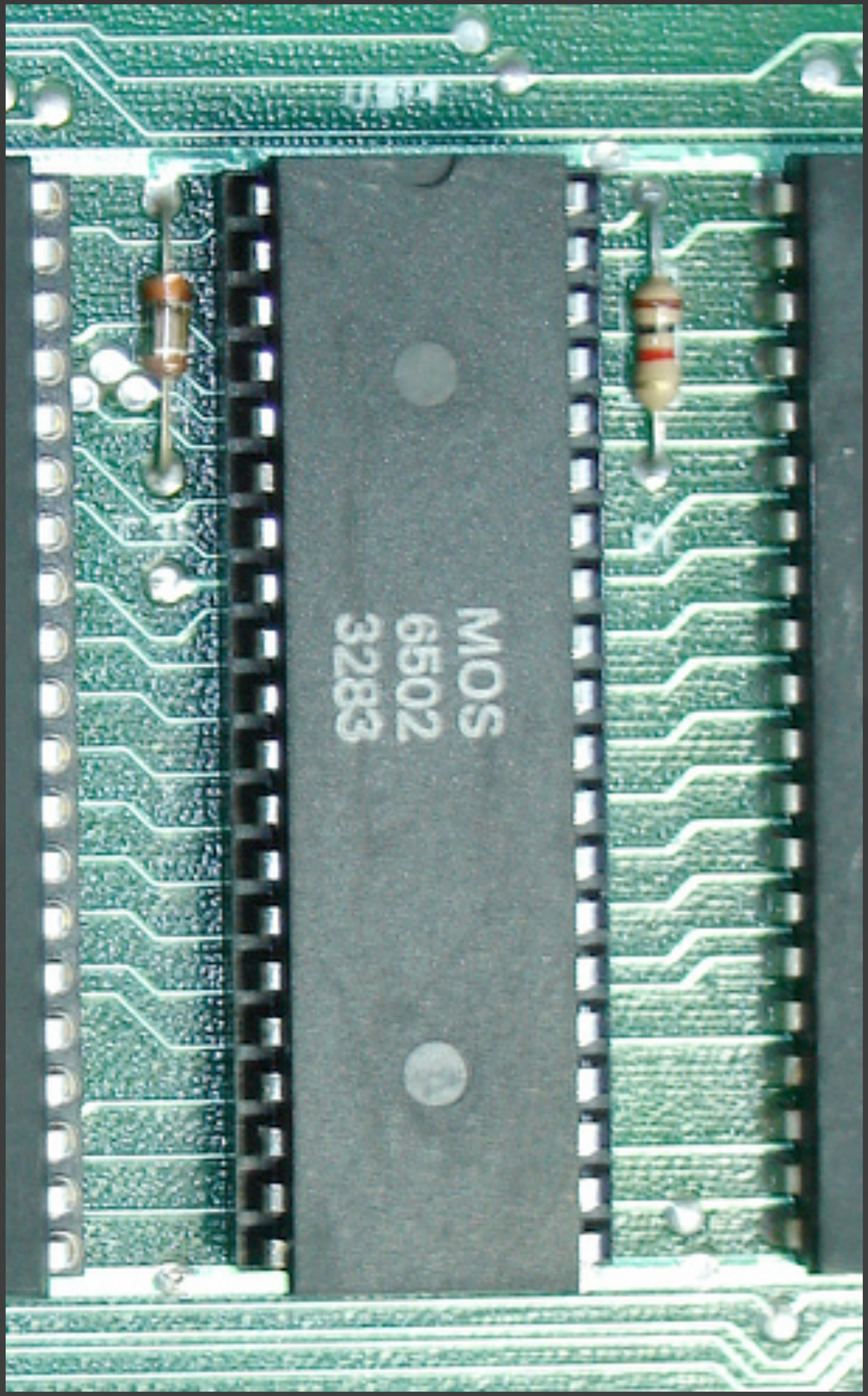
- 6502 Yesterday & Today
- Building & Programming
- Py65 Simulator Overview
- Simulation Demo
- Q&A

Maintainable
Software

History

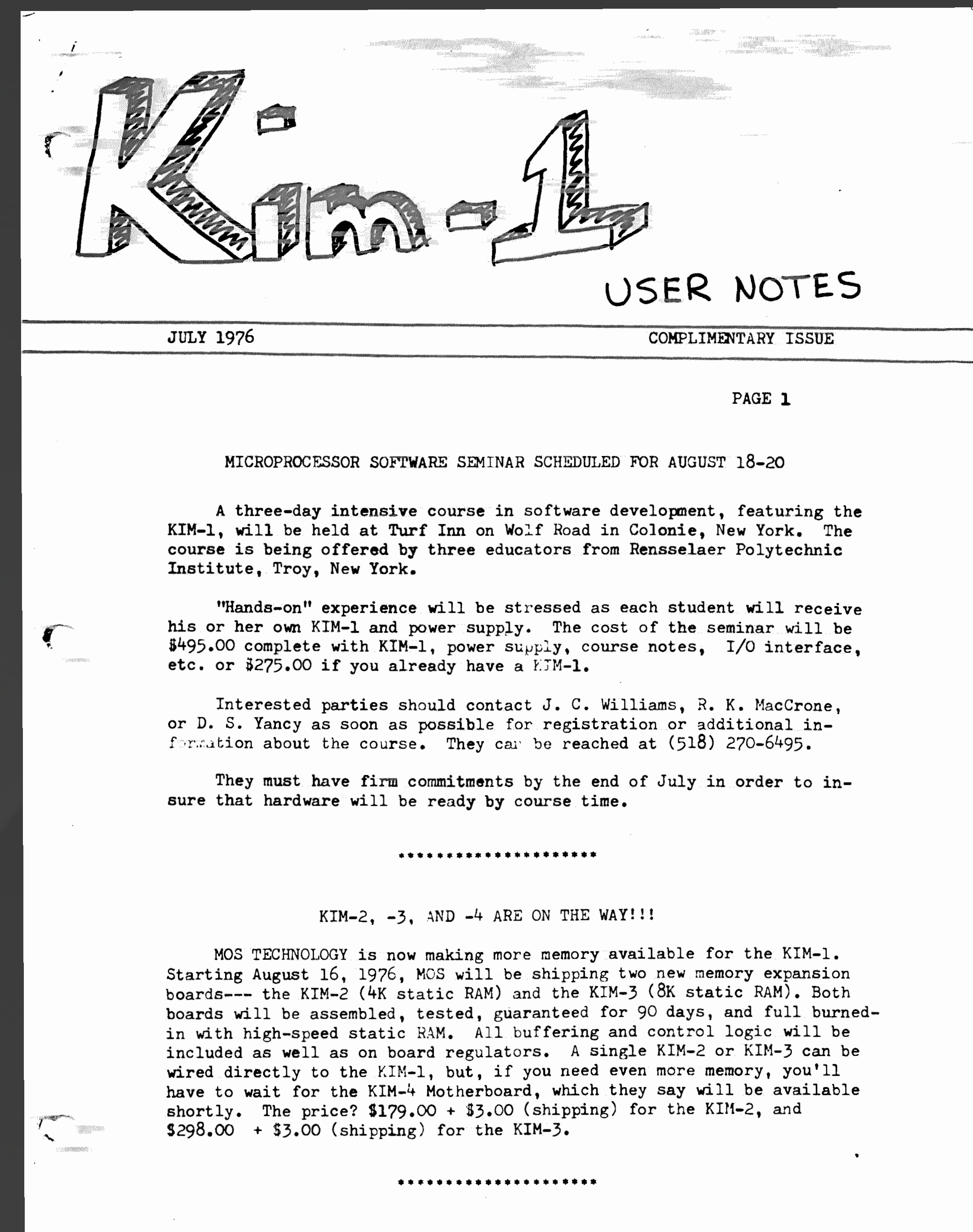
Maintainable
Software

- MOS Technology released one of the early microprocessors in the mid 1970’s.
- It was well designed.
- It was a fraction of the cost of its competitors.



Maintainable Software

- At this time, there were no home computers.
- Microprocessors made home computers possible.
- The 6502 made affordable home computers possible.



Kim-1 USER NOTES

JULY 1976 COMPLIMENTARY ISSUE

PAGE 1

MICROPROCESSOR SOFTWARE SEMINAR SCHEDULED FOR AUGUST 18-20

A three-day intensive course in software development, featuring the KIM-1, will be held at Turf Inn on Wolf Road in Colonie, New York. The course is being offered by three educators from Rensselaer Polytechnic Institute, Troy, New York.

"Hands-on" experience will be stressed as each student will receive his or her own KIM-1 and power supply. The cost of the seminar will be $495.00 complete with KIM-1, power supply, course notes, I/O interface, etc. or $275.00 if you already have a KIM-1.

Interested parties should contact J. C. Williams, R. K. MacCrone, or D. S. Yancy as soon as possible for registration or additional information about the course. They can be reached at (518) 270-6495.

They must have firm commitments by the end of July in order to insure that hardware will be ready by course time.

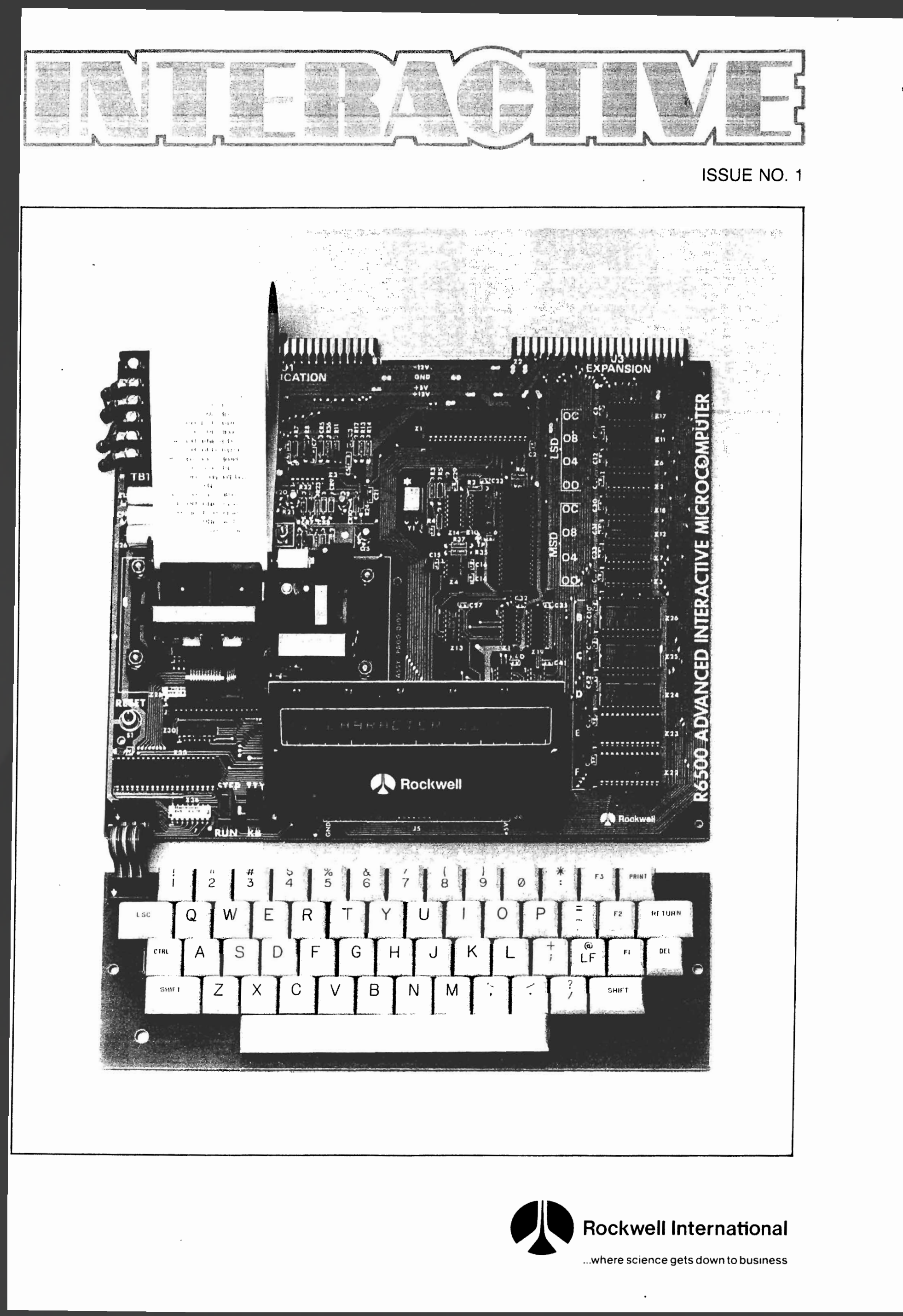
****

KIM-2, -3, AND -4 ARE ON THE WAY!!!

MOS TECHNOLOGY is now making more memory available for the KIM-1. Starting August 16, 1976, MOS will be shipping two new memory expansion boards--- the KIM-2 (4K static RAM) and the KIM-3 (8K static RAM). Both boards will be assembled, tested, guaranteed for 90 days, and full burned-in with high-speed static RAM. All buffering and control logic will be included as well as on board regulators. A single KIM-2 or KIM-3 can be wired directly to the KIM-1, but, if you need even more memory, you'll have to wait for the KIM-4 Motherboard, which they say will be available shortly. The price? $179.00 + $3.00 (shipping) for the KIM-2, and $298.00 + $3.00 (shipping) for the KIM-3.

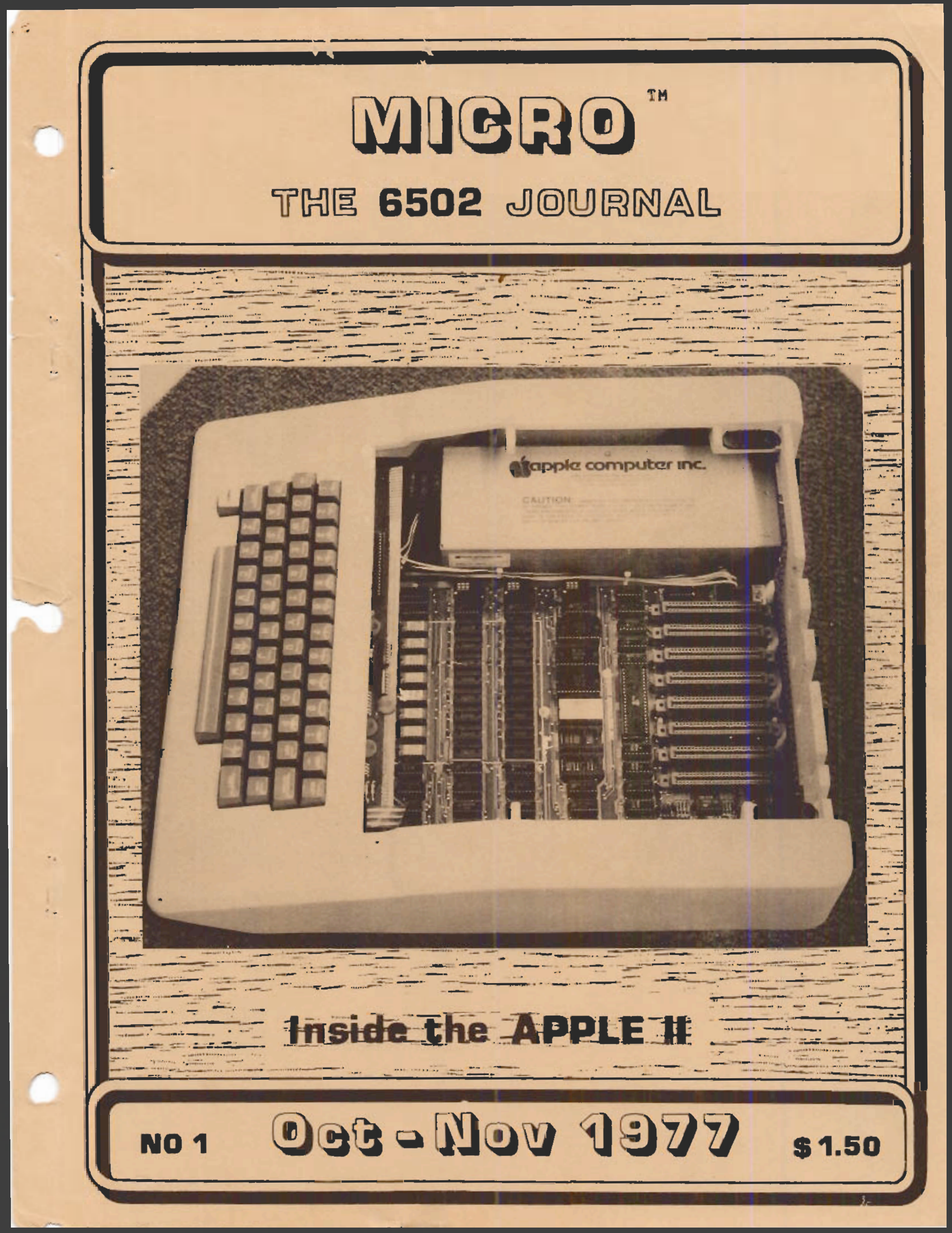
****

- With a 6502 and a few support chips, a simple computer could be made.
- People began building their own computers.
- Trainer boards like the KIM-1, SYM-1, and AIM-65 became popular.



Maintainable Software

- Self-built computers and trainers proliferated.
- They were too technical for the average person.
- Pre-built computers like the Commodore PET and Apple II made computers accessible to everyone.



Maintainable Software

By the early 1980's, 6502-based computers were everywhere. The personal computer revolution had begun.

Maintainable Software

6502 Domination

- Commodore PET, VIC-20, 64, 128...
- Apple I, II, II+, IIe, IIc, ...
- Atari 2600, 7800, 400, 800, ...
- Nintendo NES (’02), SuperNES (‘816)
- Countless others, late-70’s to mid-80’s

Maintainable Software

Embedded

- Around the mid 1980’s, 16-bit home computers began to take over the market.
- 6502-based technology was produced in huge quantities and more affordable than ever.
- 6502 applications shifted to games, embedded devices, and industrial control applications.

30+ Years of 6502

The legendary 65xx brand microprocessors with both 8-bit and 8/16-bit ISA's keep cranking out the unit volumes in ASIC and standard microcontroller forms supplied by WDC and WDC's licensees.

Annual volumes in the hundreds (100's) of millions of units keep adding in a significant way to the estimated shipped volumes of five (5) to ten (10) billion units.

With 200MHz+ 8-bit W65C02S and 100MHz+ 8/16-bit W65C816S processors coming on line in ASIC and FPGA forms, we see these annual volumes continuing for a long, long time.

- Western Design Center, Inc.

Maintainable Software

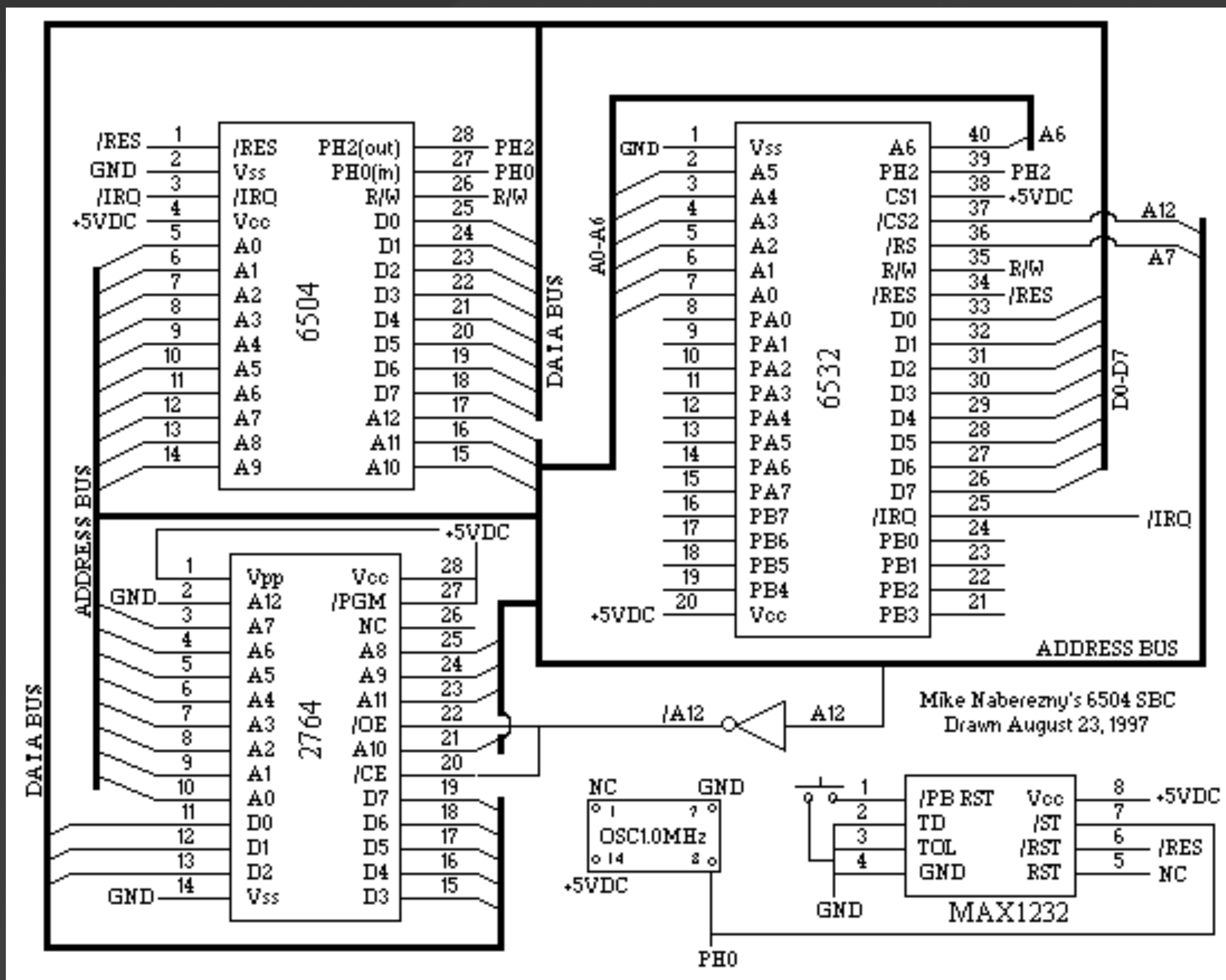
Building a Small Computer

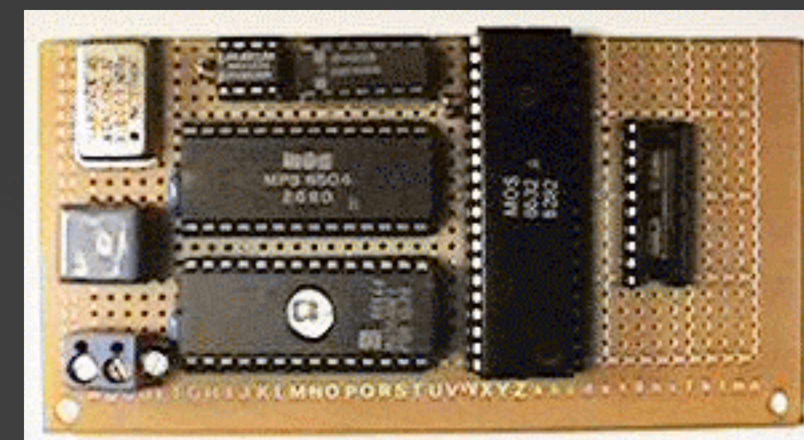
6502 System

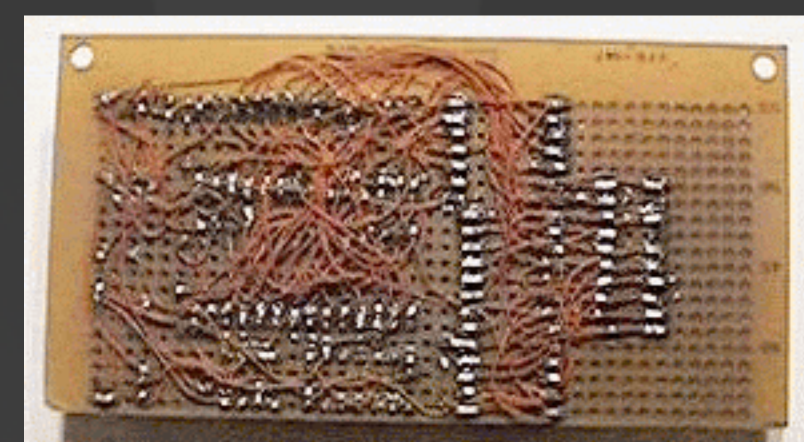
- Single 64K Address Space ($0000-$FFFF)
- RAM, ROM, I/O are all mapped into this space
- Address Lines select the address
- Data Lines hold data to read/write at the address
- Clock, RESET, decoding, other glue makes a system

Maintainable Software

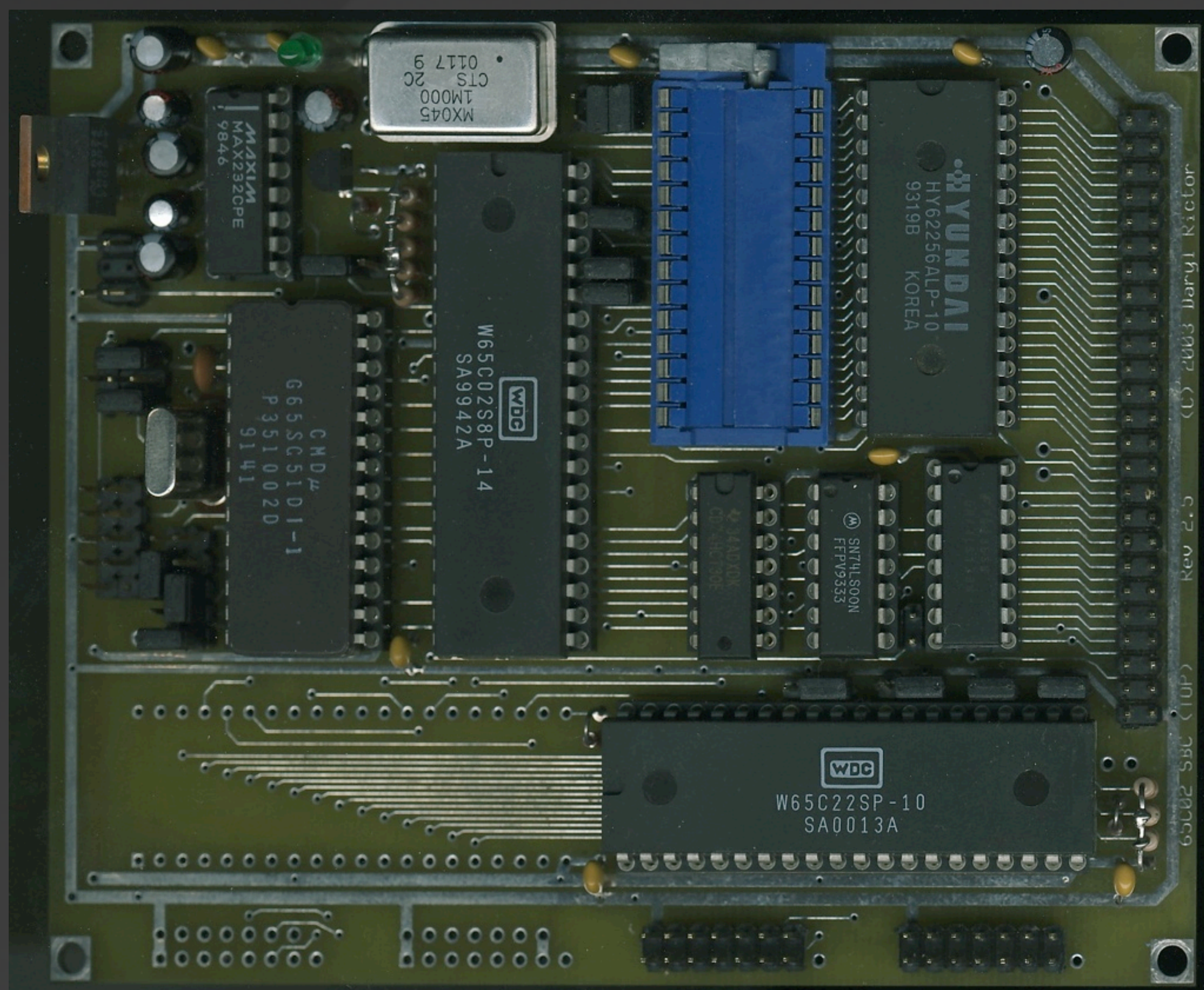
6502 System


6504
6532
2764
DATA BUS
ADDRESS BUS
A0-A6
D0-D7
/RES
GND
/IRQ
+5VDC
PH2(out)
PH0(in)
R/W
PH2
PH0
A12
A7
A6
/A12
OSC1.0MHz
NC
MAX1232
/PB RST
TD
TOL
Vcc
/ST
/RST
RST
Vpp
/PGM
/OE
/CE
Vss
Mike Naberezny's 6504 SBC
Drawn August 23, 1997





6502 System



- MPU
- Clock
- Glue Logic
- ROM
- RAM
- I/O Devices

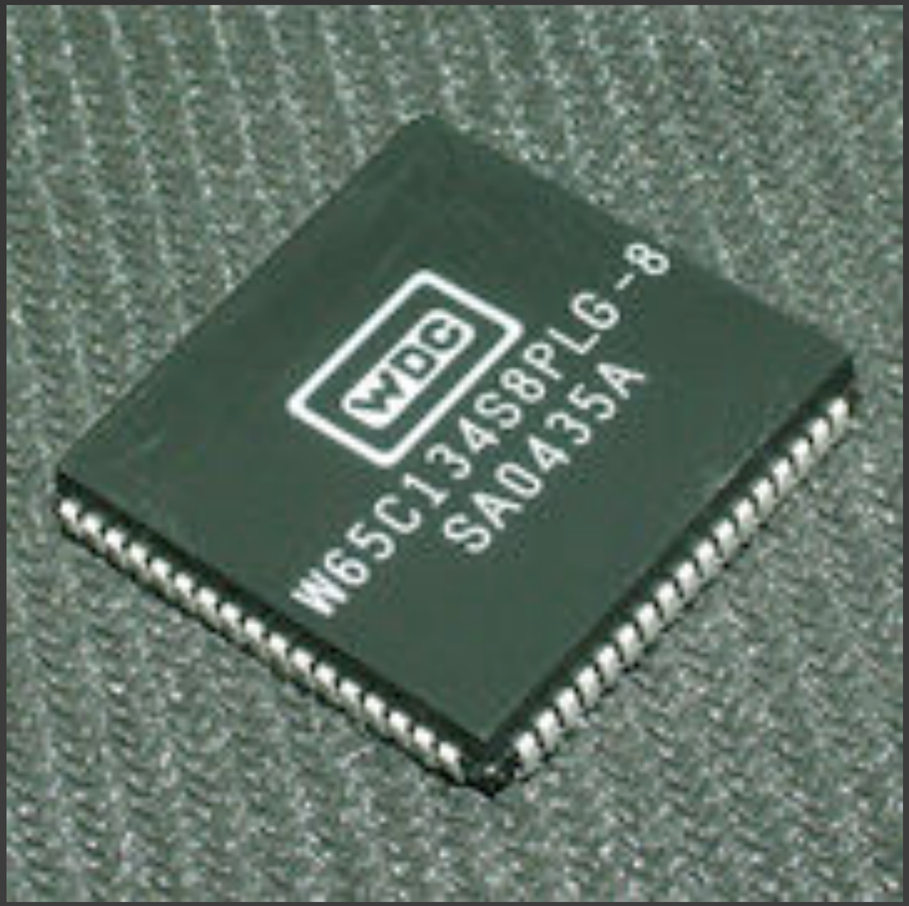
Each device is an object in Py65.

Microcomputer

- Expandable, Reconfigurable
  - Add or remove components, rewire
  - Larger, more difficult to assemble
- Observable
  - Address, data, and control lines are all accessible with logic probe or oscilloscope

Maintainable Software

Microcontroller



- Components combined into one package
- Usually fixed memory map
- Smaller, less power, etc.
- Software compatibility (same 6502 MPU core)

Maintainable Software

Microcontroller

- Fixed configuration
  - Typically not expandable
  - Smaller package, less to assemble
- Less Observable
  - Connections between internal components cannot observed (need for simulation)

Maintainable Software

Programming a Small Computer

Maintainable Software

Assembly Language

- Assembler?!
- For microcontrollers and very small computers, assembly language is still relevant and often necessary

Maintainable Software

Assembly Language

```
TOPNT = $01
* = $c000

CLRMEM = *
        LDA #$00        ;Set up zero value
        TAY             ;Initialize index pointer
CLRM1 = *
        STA (TOPNT),Y   ;Clear memory location
        INY             ;Advance index pointer
        DEX             ;Decrement counter
        BNE CLRM1       ;Not zero, continue checking
        RTS             ;Return
```

```
c000:  a9  00  a8  91  01  c8  ca  d0  fa  60
```

Maintainable Software

Problems

- Assembling even small boards is time consuming and takes some skill
- Software is developed on a PC and then downloaded into the target device
- It takes time and manual steps to download the software into the target and test it

Maintainable Software

Problems

- The software you write will often have issues during development
- Debugging these problems is difficult... often a controller hangs with no other clues as to what happened
- Software tested manually is prone to regress

Maintainable Software

Problems

- We want faster development time for very low-level software, usually assembly language.
- We want a way to test our system designs before committing to the hardware.
- We want a way to prove our software works and will continue to work when changed.

Maintainable Software

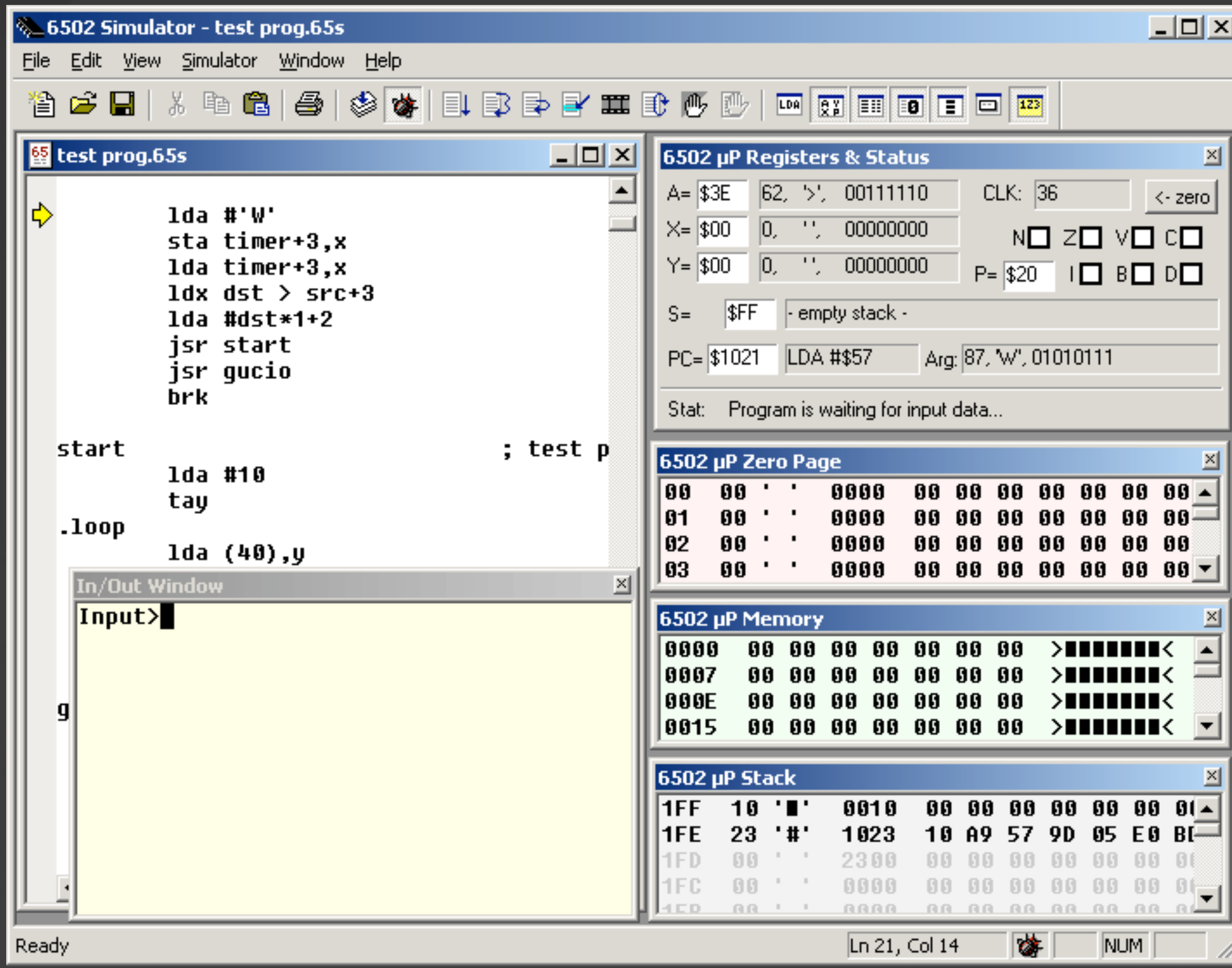
Simulation

Simulation

- Mimic the function of the hardware in a software system on a PC workstation
- Test software without downloading it into the target machine
- Allows the greatest visibility into the system

Maintainable Software

Typical Simulator



Monolithic

Maintainable
Software

Typical Simulator

- Easy to use, great learning tool, but...
  - Fixed configuration of memory map and devices; often doesn't match your target
  - Not scriptable
  - Not expandable
  - Usually not open source

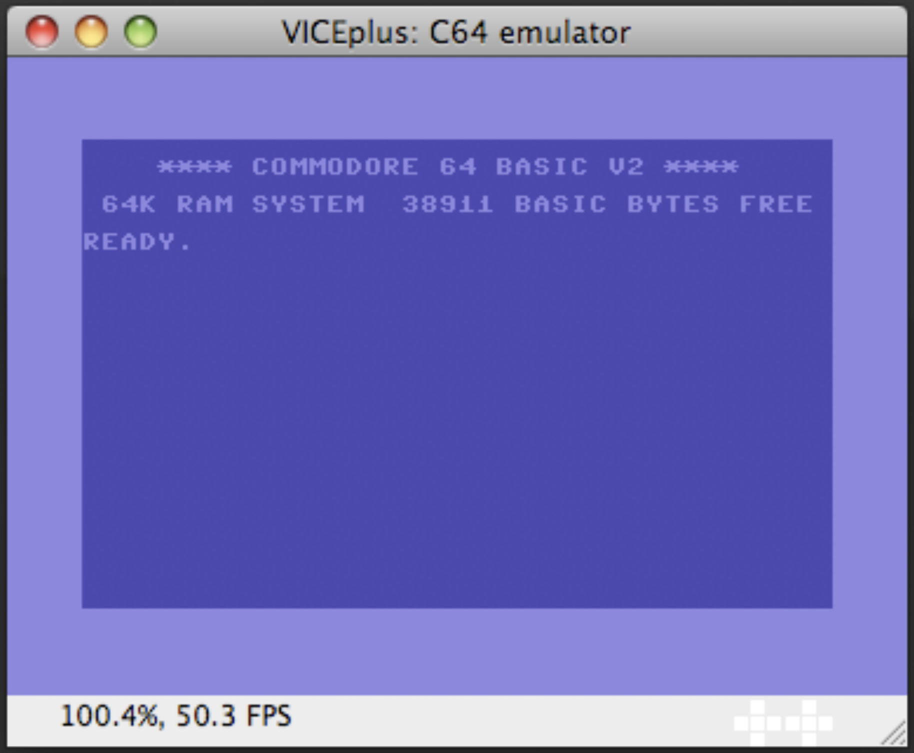
Maintainable Software

Emulators

Maintainable Software

MAME, VICE

- An emulator can be thought of as simply a simulator that runs in real-time
- MAME (Multi-Arcade Machine Emulator)
- VICE (Versatile Commodore Emulator)



Maintainable Software

MAME, VICE

- Game emulators are often more mature and advanced than tools from hardware vendors
- Provide excellent software models of many hardware building blocks (MPUs and I/O)
- Components are glued together into models of specific systems

MAME, VICE

- Typically for a different audience
- Focused on emulation (e.g. playing games) rather than as development aids
- Software is written entirely in C
- Often difficult or time consuming to make your own system models

Maintainable Software

Py65

Maintainable Software

Py65

- Focused on being an embedded development tool rather than a game emulator
- Provides building blocks for modeling systems like VICE or MAME, but less mature
- Speed is not particularly important
- Python!

Maintainable Software

Py65

- Modules organized by 65xx family device
- Objects simulate device behavior

Maintainable Software

Py65

```
Python 2.4.3 (#1, Jun 29 2008, 19:01:46)
[GCC 4.0.1 (Apple Computer, Inc. build 5367)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from py65 import mpu6502
>>> mpu = mpu6502.MPU()
>>> mpu
<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=0000>
>>> mpu.a = 0xfe
>>> mpu
<6502: A=fe, X=00, Y=00, Flags=20, SP=ff, PC=0000>
>>>
```

- Python Interactive Interpreter

Maintainable Software

Unit Tests

- Accurate emulation is much harder than it may appear even for simple microprocessors

- Py65 has ~400 unit tests for its 6502 core and test coverage is probably still < 75%

Maintainable Software

Unit Tests

```python
# LDY Immediate

def test_ldy_immediate_loads_y_sets_n_flag(self):
  mpu = MPU()
  mpu.y = 0x00
  mpu.memory[0x0000:0x0002] = (0xA0, 0x80) #=> LDY #$80
  mpu.step()
  self.assertEquals(0x0002, mpu.pc)
  self.assertEquals(0x80, mpu.y)
  self.assertEquals(mpu.NEGATIVE, mpu.flags & mpu.NEGATIVE)
  self.assertEquals(0, mpu.flags & mpu.ZERO)

def test_ldy_immediate_loads_y_sets_z_flag(self):
  mpu = MPU()
  mpu.y = 0xFF
  mpu.memory[0x0000:0x0002] = (0xA0, 0x00) #=> LDY #$00
  mpu.step()
  self.assertEquals(0x0002, mpu.pc)
  self.assertEquals(0x00, mpu.y)
  self.assertEquals(mpu.ZERO, mpu.flags & mpu.ZERO)
  self.assertEquals(0, mpu.flags & mpu.NEGATIVE)
```

Test suite verifies correct operation

Maintainable Software

Simple Loop

Maintainable Software

Simple Loop

- Since our MPU is just a Python object, we can use the interactive interpreter.
- We can also drive it with our own programs and test suites.

Simple Loop

```
*=$A000

LOOP:
  INX                ; $A000 E8
  JMP LOOP           ; $A001 4C 00 A0
```

Assembly Language

Machine Language

Maintainable Software

Simple Loop

```
*=$A000

LOOP:
  INX              ; $A000 E8
  JMP LOOP         ; $A001 4C 00 A0
```

- Load memory
- Set the PC
- Step
- Observe X

```
>>> from py65.mpu6502 import MPU
>>> mpu = MPU()
>>> mpu.memory[0xA000:0xA003] = [0xe8, 0x4c, 0x00, 0xa0]
>>> mpu
<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=0000>

>>> mpu.pc = 0xa000
>>> mpu
<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=a000>

>>> mpu.step()
<6502: A=00, X=01, Y=00, Flags=20, SP=ff, PC=a001>

>>> mpu.step()
<6502: A=00, X=01, Y=00, Flags=20, SP=ff, PC=a000>

>>> mpu.step()
<6502: A=00, X=02, Y=00, Flags=20, SP=ff, PC=a001>

>>> mpu.step()
<6502: A=00, X=02, Y=00, Flags=20, SP=ff, PC=a000>

>>> mpu.step()
<6502: A=00, X=03, Y=00, Flags=20, SP=ff, PC=a001>
```

Maintainable Software

Monitor

Maintainable
Software

Monitor

- Microprocessors often run a “machine language monitor” program.
- This is sometimes also called a “debugger”.
- Py65 has a monitor called Py65Mon.

Maintainable Software

Monitor

```
$ py65mon

Py65 Monitor

<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=0000>
.
```

- Makes common tasks like loading binaries and stepping through programs easier.
- Type “help” for commands.
- Commands mostly compatible with the monitor in the VICE emulator.

Hello World

- Py65Mon can trap writes to the memory map and display the bytes to STDOUT.
- This is enough to get us to “Hello World”

Maintainable Software

Hello World

```
*=$C000
CHAROUT=$E001

HELLO:
  LDX #$00
LOOP:
  LDA MESSAGE,X
  BEQ DONE
  STA CHAROUT
  INX
  JMP LOOP
DONE:
  RTS

MESSAGE = *
  !text "Hello, World!"
  !byte 0
```

- Program will read each character and write to $E001 until the null byte
- Py65Mon will trap the write to $E001 and send each byte to STDOUT

Maintainable Software

Hello World

```
$ py65mon

Py65 Monitor

<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=0000>
.load "hello.bin" c000
Wrote +29 bytes from $c000 to $c01c

<6502: A=00, X=00, Y=00, Flags=20, SP=ff, PC=0000>
.registers pc=c000

<6502: A=00, X=0d, Y=00, Flags=20, SP=ff, PC=c000>
.return
Hello, World!
```

- Load binary
- Set PC
- Run until RTS

Next Steps

Maintainable
Software

Next Steps

- Finish the unit test suite for the 6502 MPU core (every instruction, every mode)
- Character input trap for Py65Mon
- Run a BASIC interpreter (EhBASIC)!

Maintainable Software

Next Steps

- Two additional device models:
  - 6522 Versatile Interface Adapter (VIA)
  - 6551 Asynchronous Interface Adapter (ACIA)

Maintainable Software

Next Steps

- Add more features to Py65Mon
  - Simple assembly and disassembly
  - Select real hardware as target
- Documentation and tutorials

Maintainable Software

Q & A

Maintainable Software

Thanks!

Maintainable
Software