Microprocessors in History Part I: MOS Technology 6502

Kevin T. Woods

***Abstract*—MOS Technology changed the world. With the MCS6502 processor, this company shattered the rigid and saturated market of expensive microprocessors. No other processor made a bigger impact on the world of home computing as we know it. The mid 1970's brought about the dawn of a new era, and this chip was at the center of it all. It comes with a success story for the ages, from patent infringement law suits to landing in the hottest new toys that technological engineers could muster. You might recall some of the old systems these were found in: Apple I, Apple II, Apple III, Commodore PET, Atari and Nintendo, just to name a few. Herein, we'll take an in-depth look at this game changing processor. How it was made, who it was made for, what's in its architecture and design, and why it had such success will all be addressed in this first of two papers on historical processors.**

***Index Terms*—6502, Apple I, Apple II, Central Processing Unit, microprocessors, processor, MOS Technology**

Fig. 1. MOS Technology MPS6502 / 6502. Courtesy of CPU-World.com

I. Introduction

Let's go all the way back to 1975, to a new start-up company called MOS Technology. They made the 6502 processor, most notably known for its use in the original Apple computers, early Atari game systems, Commodores, and the Acorn Microcomputer. This was the age when personal computers were a far-fetched pipe dream and the only viable use for a computer was in the office. Luckily, that dream would soon be realized. Of course, much of the credit in reaching that phenomenal dream of personal home computers goes to the innovation of Steve Jobs, Bill Gates and their many gifted associates and counterparts in the decade that followed.

Prior to this processor, computers had generally come in the form of large machines used solely for business purposes, with no built-in visual displays. I am eternally grateful for the landmarks achieved during this time as they laid the ground work that shaped the world as we know it today, and for the work that I will be doing throughout my career. Much of that ground work that was built around this simple yet powerful processor.

II. Processor History

In the early days of chip design, everybody was getting patents on anything they developed. Only a couple years after Intel came out with the first 8-bit processor, other companies started up or joined in the race. MOS Technology was one of those newcomers, and so, to, comes a not-so-frivolous story of patent infringement with MOS Technology's 6500 family of microprocessors.

Fig. 2. MOS 6502 Saves more Money [2].

A. Evolution from prior designs

Their 6502 processor made its mark on the computing world within a few short months of its release, and Motorola noticed. After only two months, they claimed that its design infringed on some of their patents and sued MOS Technology. The 6502 apparently comes from some of the same designs patented by Motorola and used in the production of their MC6800 a year before the 6502 launched. This was possible because the inventors listed on the patents were Motorola employees that left and then joined MOS Technology to create

Paper due 9 Oct, 2015, submitted to and sponsored by Dr. Adam W. Lewis, professor of Computer Science at Athens State University, Athens, AL 35611 USA.

K. T. Woods is an undergraduate student of Computer Science at Athens State University, Athens, AL 35611 USA (e-mail: kwoods3@my.athens.edu)

the 65xx processor family. The major share-holder decided not to fight it, and eventually both companies agreed to cross-license their work as needed.

B. *Targeted market*

Interestingly, while many semiconductor companies targeted only businesses that might make large orders, MOS Technology had the idea to target the individual and offer the general purpose 6502 at an extremely low cost of just $25 at the WESCON ‘75 conference, as shown in Figure 2. This strategy worked well for the company and many individuals caught the 6502 fever. Engineers and hobbyists around the country all wanted to get their hands on one as it was touted as easy to use, and if you had a KIM-1, up to 4 KB of extra memory could be readily adapted to be used in the KIM-1.

C. *Preferred market*

Within a year, the 6502 had gained such notoriety that a then newcomer and garage start-up company, Apple Computer, Inc., used it to launch their flagship Apple I computer. They stayed with the chip for the Apple II and the Apple III. Nintendo, Commodore, BBC, and Atari all found uses for it as well and sales skyrocketed.

III. Processor Design

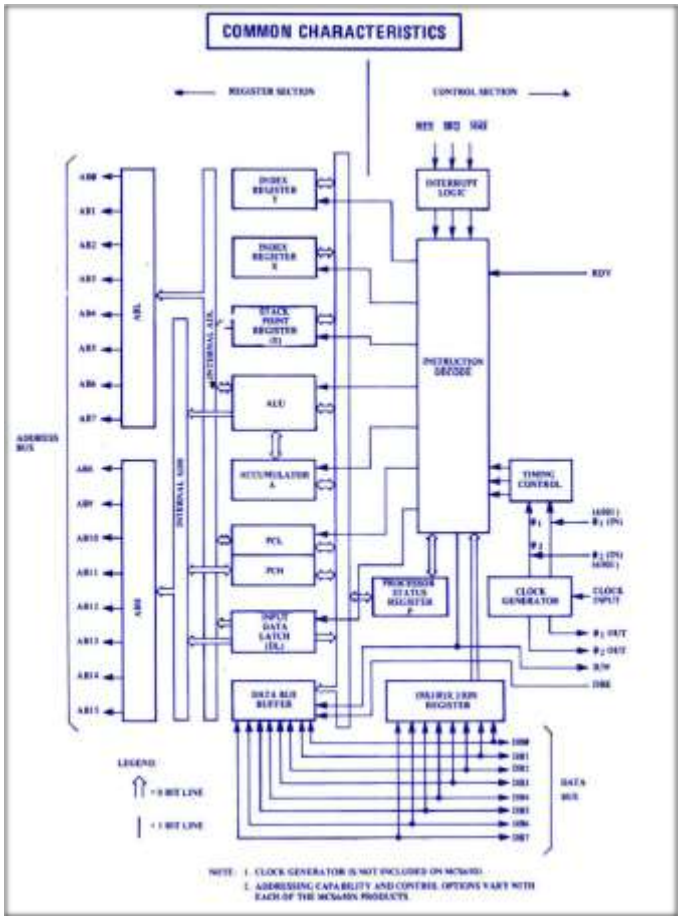

Fig. 3. MCS6501 – MCS6505 Internal Architecture [7].

A. *Chip Packaging*

The MCS6502 came in a 40 pin package, with an on-the-chip clock, that when combined with the MC6800 bus compatibility of the processor, eliminated the need to have a two phase 5 volt clock input. Because of that, the chip was able to be driven by a single TTL level input clock, an RC time base, or a Crystal time base.

B. *Instruction set Architecture*

There were 56 listed instructions for the MCS6502 processor, included Rotate Right (ROR). The lack of a working ROR was a major flaw of the MCS6501. Single instruction operations were allowed by having a SYNC line output that signaled every time an operation code was fetched. There was also a Ready signal that could be used in single cycle executions.

C. *Memory Hierarchy*

There were very few registers in this processor. The accumulator, X index, Y index, program counter, and stack pointer made up the entire register list. No cache memory existed at the time this processor was in mainstream use.

D. *Addressing Modes*

With 65K addressable bytes of memory, the 6502 was capable of many addressing modes. First, we have the ACCUMULATOR addressing, which was a one byte instruction operating on the accumulator. IMMEDIATE addressing contained the operand in the second byte of the instruction and had no further memory references. With IMPLIED addressing, the address containing the operand was implicitly given in the operation code. RELATIVE addressing provided the destination for branch instructions.

In ABSOLUTE addressing, the instruction’s second byte contained low order bits and the third byte had the high order bits, allowing all 65K bytes of memory to be addressed. INDEXED ABSOLUTE addressing, used in conjunction with the index register, provided a way to use X, Y indexing, sometimes referred to as “Zero Page, X” and “Zero Page, Y” where the effective address was calculated by adding the contents of the second byte to the index register’s value.

ZERO PAGE addressing had instructions that would enable shorter code and faster execution times due to only fetching the second byte of the instruction while assuming a zero high order byte. INDEXED ZERO PAGE, used in conjunction with the index register, provided a way to use X, Y indexing, sometimes referred to as “Absolute, X” and “Absolute, Y” where the effective address was calculated by adding the contents of X or Y to the address in the second and third bytes.

INDEXED INDIRECT (Indirect, X) added the second byte of the instruction to the value in the X register and discarded any carry. The resulting value pointed to a page zero location in memory that contained the low order bits of the effective address. The high order bits of the effective address were contained in the very next page zero memory location.

INDIRECT INDEXED (Indirect, Y) was where the second instruction byte pointed to a page zero location that would then be added to the Y register value to obtain the low order bits of the effective address. Further, adding the resulting carry to the next zero page memory location gave the high order bits of the address.

Lastly, we have the ABSOLUTE INDIRECT addressing, which is where the second byte of the instruction contains the low order bits of a memory location and the high order bits are in the third byte of the instruction. The contents of that memory location were the low order bits of the effective address, and the next memory location had the high order bits of the effective address. These sixteen bits were loaded into the program counter register.

E. Assembler Syntax

See Figure 4 for an example of the 6502 assembly language.

```
Here's another good number from Stan Ockers which will prove useful when giving KIM
demonstrations at your next club meeting or maybe for your family when they begin to
wonder what happened to the household budget.

                ***** GET PROGRAMS *****          by Stan Ockers

    THIS PROGRAM ALLOWS YOU TO CREATE A LIBRARY OF OTHER PROGRAMS
WHICH ARE THEN LOADED INTO EXTRA MEMORY AREA.  WHEN RUN, THE
PROGRAM WILL LIST THE AVAILABLE PROGRAMS ON A TVT ALONG WITH
ASSOCIATED NUMBERS (0-F).  PRESSING THE ASSOCIATED NUMBER ON THE
KIM KEYPAD WILL CAUSE THE PROGRAM TO BE TRANSFERRED INTO THE
REGULAR KIM MEMORY WHERE IT WILL BE STARTED.  THE STARTING LOCATION
OF THIS PROGRAM IS LOADED INTO ITFA AND ITFB SO THAT PRESSING
THE STOP BUTTON ON THE KIM WILL DISPLAY THE CHOICES AGAIN.

2000    A0 00       LDY #$00        INIT. INDEX
  02  B9 E0 20 MORE LDA 2030,Y      GET CHARACTER
  05    C9 FF       CMP #$FF        FINISHED?
  07    F0 0A       BEQ WAIT        YES
  09    84 E6       STY 00E6        SAVE INDEX
  0B  20 A0 1E      JSP OUTCH       OUTPUT CHAR.
  0E    A4 E6       LDY 00E6        RETURN INDEX
  10    C8          INY             INC. INDEX
  11    D0 LF       BNE MORE        UNCOND. JUMP
  13  20 1F 1F WAIT JSP SCANDS      WAIT FOR KEY
  16    F0 FB       BEQ WAIT        NO KEY DOWN
  18  20 6A 1F      JSP GETKEY      WHAT KEY
  1B    C9 15       CMP #$15        VALID KEY?
  1D    10 F4       BPL WAIT        NO
  1F       A8       TAY             USE AS INDEX
  20    A2 07       LDX #$07        8 VALUES FROM TABLE
  22  B9 50 20 TABL LDA 2050,Y      GET POINTER
  25    95 E7       STA 00E7,X      STORE IT
  27       98       TYA             UPDATE INDEX
  28       18       CLC
  29    69 10       ADC #$10
  2B       A8       TAY
  2C       CA       DEX             MORE VALUES?
  2D    10 F3       BPL TABL        YES
  2F    A6 EA       LDX 00EA        PAGES TO TRANSFER
  31    F0 14       BEQ REMA        LESS THAN ONE PAGE
  33    A0 FF       LDY #$FF        INDEX FOR NEXT PAGE
  35  B1 EB MORE    LDA (00EB),Y    GET BYTE
  37    91 ED       STA (00ED),Y    MOVE IT
  39       88       DEY
  3A    C0 FF       CPY #$FF        LAST BYTE?
  3C    D0 F7       BNE MORE        NO
  3E    E6 EC       INC 00EC        UPDATE PAGES
  40    E6 EE       INC 00EE
  42       CA       DEX             ANY MORE PAGES?
  43    30 06       BMI PGRM        NO
  45    D0 EE       BNE MORE        YES
  47    A4 E9 REMA  LDY 00E9        PART OF A PAGE
  49    D0 EA       BNE MORE        UNCOND. JUMP
  4B  6C E7 00      JMP (00E7)      INDIRECT JUMP

VALUES MUST BE PLACED IN THE FOLLOWING TABLE TO INDICATE WHERE
LIBRARY PROGRAMS START, WHERE THEY ARE LOCATED AND HOW MANY
BYTES THEY CONTAIN.  THE VALUES FOR PROGRAM #0 ARE STORED IN
2050,2060,2070 ETC. THE VALUES FOR PROGRAM #1 ARE STORED IN
2051,2061 ETC. UP TO 16 PROGRAMS CAN BE REFERENCED.

2050-205F  TO LOCATION, HIGH (PAGE IN KIM-1 WHERE PROGRAM IS TO
                                BE LOADED)
2060-206F  TO LOCATION, LOW (LOW VALUE TO GO WITH ABOVE)
2070-207F  FROM LOCATION, HIGH (PAGE WHERE PROGRAM IS FROM)
2080-208F  FROM LOCATION, LOW
2090-209F  # OF PAGES TO TRANSFER
20A0-20AF   # BYTES IN ADDITION TO THAT NUMBER OF PAGES
20B0-20BF  STARTING LOCATION OF PROGRAM, HIGH
20C0-20CF  STARTING LOCATION OF PROGRAM, LOW
```

Fig. 4. Example 6502 assembly program, written by Ockers [7].

Software support for this processor was limited only by the imagination as times were changing and home computing and gaming were becoming a viable option. The 8-bit processing power enabled simple real-time graphics handling and more complex multi-functioning programs. With this architecture, we saw the first versions of video game systems, office and document software, and interactive GUI based operating systems.

IV. Analysis

With the vast majority of systems that utilized the MOS Technology 6502 processor reaching into the homes of millions of people, this processor pioneered the greatest progress in computing history and changed the world.

It had a rather slow clock cycle but since was designed with an elegant but simple two-phase clock, instructions could complete in about half the clock cycles as other processors. This made the 6502 competitive even with its slower speeds. The simplistic design heavily attributed to the lower costs of building the chips, which also made it more desirable to consumers.

There were some variations in the 6500 family of processors, mostly with fewer pins and varying amounts of addressable memory. Some even had audio processing capabilities as used in the Nintendo. The NMOS 6502 preceded the CMOS 65C02, which had much of the same architecture but required less power and could support higher speeds and new instructions. Other derivatives came in the form of 16-bit processors down the road, such as were used in the Super Nintendo.

The early versions of the 6502 had a few issues; most notably the rotate right (ROR) instruction failed to produce expected results. It actually acted more like a rotate left. MOS Technology was aware of the bug and intentionally left the instruction out of the design documentation until it was addressed in later versions, starting in very late 1975. A memory indirect JMP instruction that ended in hex xxFF would not jump to the correct address, which was never fixed in any of the NMOS versions of the chip but was corrected in its CMOS brothers.

References

[1] "Motorola Sues MOS Technology," *Microcomputer Dig.*, vol. 2, no. 6, pp. 11, Dec. 1975. [Online]. Available: http://bitsavers.trailing-edge.com/pdf/microcomputerAssociates/Microcomputer_Digest_v02n06_Dec75.pdf

[2] "MOS 6502 the second of a low cost high performance microprocessor famnily," *Computer*, vol. 8, no. 9, pp. 38-39, Sept. 1975. Ad. doi: 10.1109/C-M.1975.219074

[3] *Preliminary data sheet for MCS6501 – MCS6505 microprocessors*, MOS Technology Inc., Norristown, PA, 1975. [Online]. Available: http://archive.6502.org/datasheets/mos_6501-6505_mpu_preliminary_aug_1975.pdf. Accessed on: Sept. 21, 2015.

[4] J, Aycock, "Applied Computer History: Experience Teaching Systems Topics through Retrogames," in *ITiCSE 2015*, Vilnius, Lithuania, 2015, pp. 105-110, doi: 10.1145/2729094.2742583

[5] H. G. Baker, "the COMFY 6502 Compiler," ACM SIGPLAN Notices, vol. 32, no. 11, Nov. 1997. [Online]. doi: 10.1145/270941.270947

[6] A. Ferruzzi, "Inside the Apple II," *MICRO the 6502 J.*, no. 1, pp. 9, Oct. 1977. [Online]. Available: http://archive.6502.org/publications/micro/micro_01_oct_1977.pdf. Accessed on: Sept. 21, 2015.

[7] A. Ferruzzi, "Rockwell International and the 6502," *MICRO the 6502 J.*, no.1, pp. 10, Oct. 1977. [Online]. Available: http://archive.6502.org/publications/micro/micro_01_oct_1977.pdf. Accessed on: Sept. 21, 2015.

[8] G. James, B. Silverman and B. Silverman, "Visualizing a Classic CPU in Action," in *SIGGRAPH 2010*, Los Angeles, CA, 2010, art. 26.

[9] S. Ockers, "Get programs," *KIM-1/6502 User Notes*, no. 6, pp. 5, Jul. 1977. [Online]. Available: http://archive.6502.org/publications/6502notes/6502_user_notes_6.pdf. Accessed on: Sept. 21, 2015.

[10] B. Salzsieder, "Cheap Memory for the KIM-1," *MICRO the 6502 J.*, no. 1, pp. 3-4, Oct. 1977. [Online]. Available: http://archive.6502.org/publications/micro/micro_01_oct_1977.pdf. Accessed on: Sept. 21, 2015.

[11] M. Steil, "Measuring the ROR bug in the Early MOS 6502," *Pagetable.com category archives: 6502*, Sept. 2010. [Online]. Available: http://www.pagetable.com/?p=406. Accessed on: Oct 9, 2015.

[12] G. Shvets, "Differences between 65xx, 65Cxx, and 65SCxx microprocessors," *CPU World*, last modified July 14, 2015, http://www.cpu-world.com/info/6502/65xx_65Cxx_65SCxx_differences.html