AN IBM 370 BC MODE SIMULATOR FOR A FIRST COURSE IN OPERATING SYSTEMS

John C. Peck, Clemson University

ABSTRACT

An IBM 370 basic control mode simulator was developed as an outgrowth of student projects in an operating systems course over several semesters. The programming was performed in Assembler language to enhance performance and allow for online symbolic access to simulated hardware components.

The design of the simulator along with the organization of a typical instruction interpretation subroutine is described. A simple program execution using the IBM TSO facility is included to illustrate the manner in which students interact with the system. Finally the design for a typical operating system undertaken as a team project is presented.

INTRODUCTION

A first course in operating systems can be taught at several different levels ranging from a very theoretical to a very practical. The course at Clemson University is somewhere between these two extremes, although perhaps closer to the practical. Students enrolled in the course have successfully completed a course in IBM 370 Assembler language and a course in Systems Programming with a strong emphasis in data management and supervisor services.

Stratgies presented for the management of resources are reinforced through the use of a "hands-on" approach in which students use an IBM 370 basic control mode simulator (non-virtual storage) to implement and evaluate basic algorithms.

Reference material for the course consists of a primary text - Operating Systems by Madnick and Donovon, various IBM technical manuals, and numerous articles taken from current literature. Normal enrollment is five senior students and five first year graduate students.

HISTORY OF DEVELOPMENT

Courses in operating systems at Clemson University for many years used the University mainframe computer during early morning hours to test simple interrupt processing programs in a stand alone environment. In addition to extreme inconvenience for students other disadvantages were:

- time consuming debugging due to necessity to load the standard operating system and assembler between test runs
- poor utilization of resources during periods of dedicated usage
- only small groups of students could be served

During the fall semester of 1974 the number of students enrolled in the course reached a level which made dedicated hardware testing extremely difficult. As a result the class undertook the design of a machine simulator as a project so that future classes might have better facilities for machine interface. Although little code was produced during the semester, several approaches were investigated in detail and general program organization standards were established.

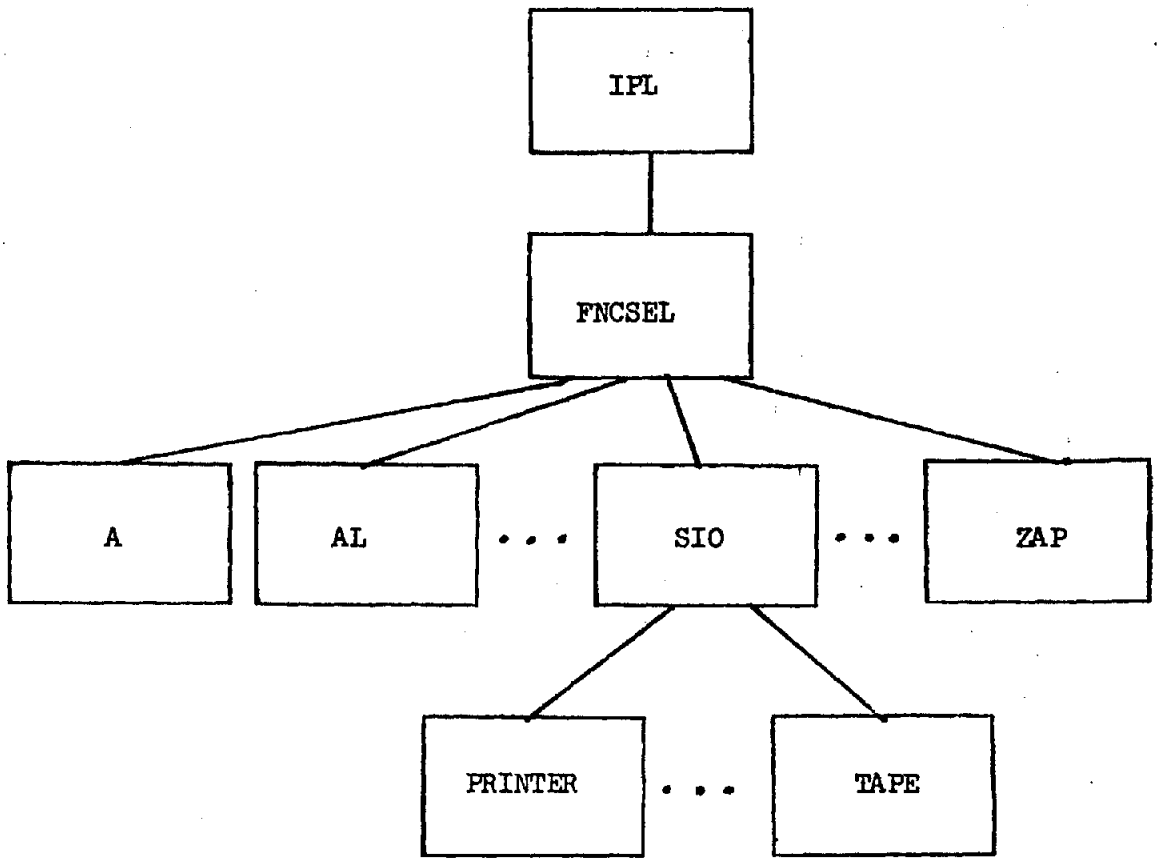
One of the more enterprising students of that semester consolidated the ideas discussed during the design phase and undertook the implementation as a one hour master's degree project. Although this work did not produce a fully useable product, it did provide an excellent foundation for continued development of the complete simulator.

The fall 1975 semester class divided the remaining work and completed the coding and testing. Several simple operating systems completed in prior semesters using "real" hardware were successfully run on the simulated hardware as final system tests.

SIMULATOR DESIGN

An IBM 370 with 32K main storage, one card reader, one printer, and four tape drives was simulated using instruction timings corresponding to execution on a model 158. Only basic control mode was simulated so that only real storage programming is possible. All programming for the simulator was done in Assembler language so that symbolic execution using the IBM TSO TEST facility would be possible

Execution of the simulator begins with the setting of a device address for the initial program load (IPL) through the reading of an address record from a special IPL file. The IPL is then simulated by executing an "implied" start I/O (SIO) instruction to the IPL address using a channel command word (CCW) which reads twenty-four bytes of data into low memory and command chains to a CCW at location 8. Following the I/O operation the program status work (PSW) is loaded from location 0 and execution begins. The design of the simulator is modular as indicated below.



A function selector module (FNCSEL) uses the PSW next instruction address to select an instruction for execution: the opcode of this instruction is used to determine the address of a subroutine which simulates the execution of the instruction. The subroutine adjusts the PSW so that when control is returned to FNCSEL a new instruction will be selected.

A typical subroutine for simulating an instruction is shown below.

```
CVB      RXINIT 8                  PERFORM LINKAGE SETUP
         RXSET                     ISOLATE OPERANDS - PUT IN R3 & R4
         PSWAD 4                   SET INSTR LNG CODE IN PSW
         CLCLK 9003                INCREMENT CLOCK FOR INSTR TIMING
         ADRES PARALIST,RETURN     CHECK FOR ADDRESS EXCEPTION
         DATA  PARALIST,RETURN     CHECK FOR DATA EXCEPTION
*****CHECK FOR OUT-OF-RANGE
         CP    0(8,R3),MAX         TOO BIG?
         BH    ERR
         CP    0(8,R3),MIN         TOO SMALL?
         BNL   GOOD
ERR      PGMINT 0009,10,RETURN     GEN.RATE PROG INTRPT CODE 9
*****
GOOD     CVB   R8,0(R3)            CHANGE DATA FROM SIMULATED MEMORY
         ST    R8,0(R4)            PUT BACK INTO SIMULATED REGISTER
RETURN   STCP                      RETURN TO FUNCSEL
MAX      DC    PL8'2147483647'
MIN      DC    PL8'-2147483648'
         LTORG
         IBM370                    DSECT FOR SIMULATED MACHINE
         END
```

A set of macro definitions is used to check for program exceptions so that implementation of most instructions is a simple matter of coding a series of macros and finally coding an instruction to produce the same effect as the simulated instruction.

When each I/O routine is executed a simulated clock associated with each device is set to a value which exceeds the simulated time-of-day (TOD) clock by an amount equal to the time required to complete the I/O. When control is returned to the FNCSEL, each clock is checked to determine if the TOD clock exceeds any device clock and an interrupt should be simulated. An inactive (available) device is indicated by a high value setting of its clock.

The card reader and printer are simulated with sequential files while the tapes are simulated with direct access files and EXCP coding. Tape marks are simulated by using key fields on a direct access record. Reverse reading is performed by retaining count field information, executing a 'search ID equal' followed by a 'read data' channel program, and moving the data to the simulated memory address using the residual byte count in the channel status word.

SAMPLE CONVERSATIONAL SESSION

The simulator will operate in either batch or online mode; however, execution with IBM's TSO TEST facility is much superior to a batch since the student can react to machine and program conditions in a manner very similar to operation of the real machine. Symbolic debugging is possible with TEST so that access to various simulated hardware components can be made by name rather than address. The complete simulated machine is symbolically defined by:

```
          MACRO
          IBM370
IBM370    DSECT
PSW       DS    XL8
          ORG   PSW
CHMASKS   DS    XL1
PROKEY    DS    XL1
INTCODE   DS    XL2
PRGMASK   DS    XL1
INSTADDR  DS    XL3
CLOCK     DS    XL8
RCLOCK    DS    XL8
T1CLOCK   DS    XL8
T2CLOCK   DS    XL8
T3CLOCK   DS    XL8
T4CLOCK   DS    XL8
PCLOCK    DS    XL8
GPR       DS    16XL4
          ORG   GPR
GPR0      DS    XL4
GPR1      DS    XL4
GPR2      DS    XL4
GPR3      DS    XL4
GPR4      DS    XL4
GPR5      DS    XL4
GPR6      DS    XL4
GPR7      DS    XL4
GPR8      DS    XL4
GPR9      DS    XL4
GPR10     DS    XL4
GPR11     DS    XL4
GPR12     DS    XL4
GPR13     DS    XL4
GPR14     DS    XL4
GPR15     DS    XL4
FPR       DS    XL32
MPROTKEY  DS    XL16
MEMORY    DS    XL32767
          ORG   MEMORY
IPLPSW    DS    XL8
IPLCCW1   DS    XL8
IPLCCW2   DS    XL8
EXTOPSW   DS    XL8
SVCOPSW   DS    XL8
PGMOPSW   DS    XL8
MCHKOPSW  DS    XL8
IOOPSW    DS    XL8
CSW       DS    XL8
CAW       DS    XL4
          DS    XL4
INTERTMR  DS    XL4
          DS    XL4
EXTNPSW   DS    XL8
SVCNPSW   DS    XL8
PGMNPSW   DS    XL8
MCHKNPSW  DS    XL8
IONPSW    DS    XL8
          MEND
```

A typical session using TSO is shown below.

PROGRAM

```
PROGRAM
800 0520                  START   BALR  2,0
802 41302046                      LA    3,CCW
806 50300048                      ST    3,X'48'
80A F342203D203A          LOOP    UNPK  NUM(5),PNUM(3)
810 96F02041                      OI    NUM+4,X'F0'
814 9C00000E                      SIO   X'00E'
818 47702012                      BNZ   *-4
81C 82002066              WAIT    LPSW  WAITPSW
820 91040044                      TM    X'044',X'04'
824 47E0201A                      BNO   WAIT
828 FA20203A2042                  AP    PNUM(3),ONE(1)
82E F921203A2043                  CP    PNUM(3),TEN(2)
834 47D02008                      BNH   LOOP
838 82000A6E                      LPSW  STOPPSW
83C 00000C                PNUM    DC    PL3'0'
83F 4040404040            NUM     DC    CL5' '
```

```
844 1C                  ONE         DC    P'1'
845 010C                TEN         DC    PL2'10'
847 C3D6E4D5E3C5D97A40  TITLE       DC    C'COUNTER: '
850 0100084780000008    CCW         CCW   1,TITLE,X'80',8
853 0100000090000003                CCW   1,0,X'90',3
860 0100083F00000005                CCW   1,NUM,0,5
868 FF02000000000000    WAITPSW     DC    X'FF02000000000000'
870 0002FFFF00000000    STOPPSW     DC    X'0002FFFF00000000'
                                    ORG   X'78'
078 FF0000000000000820              DC    X'FF0000000000000820'
                                    END
```

EXECUTION SESSION (operator entry shown by →)

```
→   exec ibm370 list

    ALLOC FI(TERM1) DA(*)
    ALLOC FI(TERM2) DA(*)
    ALLOC FI(CARDIN) DA(*)
    ALLOC FI(TAPE1) DA(*)
    ALLOC FI(PRINT) DA(*)
    ALLOC FI(ISTEPI) DA(*)
    ALLOC FI(ISTEPO) DA(*)
    TEST 'CPSC423.PGMLIB.LOAD(IBM370)'
    Q MEMORY
    +800=X'05204130204E50300048F342203D203A96F020419C00000E47702012'
    +81C=X'820206691040044447E0201AFA20203A2042F921203A204347D02008'
    +838=X'8200206E00000C40404040401C010CC3D6E4D5E3C5D97A4001000084780000008'
    +858=X'0100000090000003010008 3F00000005FF02000000000000000002FFFF00000000'
    +78=X'FF0000000000000820'
    PSW=X'FF0000000000000800'
    TEST

→   list +800 i 1(60)

      +800    BALR  2,0
      +802    LA    3,78(0,2)
      +806    ST    3,72(0,0)
      +80A    UNPK  61(5,2),58(3,2
      +810    OI    65(2),X'F0'
      +814    SIO   14(0)
      +818    BC    7,18(0,2)
      +81C    LPSW  102(2)
      +820    TM    68(0),X'04'
      +824    BC    14,26(0,2)
      +828    AP    58(3,2),66(1,2)
      +82E    CP    58(3,2),67(2,2)
      +834    BC    13,8(0,2)
      +838    LPSW  110(2)
    TEST

→   go

    ENTER IPL ADDRESS OR 'N' FOR NO IPL

→   n

    DO YOU WISH TO ISTEP?   (Y/N)

→   y
```

```
FF00000000000800

FF00000040000802

FF00000080000806

FF00000080000808

FF000000C0000810
→ ½                (break key)
TEST

→ istep=c'N'


TEST

→ go


COUNTER:   00000
COUNTER:   00001
COUNTER:   00002
COUNTER:   00003
COUNTER:   00004
COUNTER:   00005
COUNTER:   00006
COUNTER:   00007
COUNTER:   00008
COUNTER:   00009
COUNTER:   00010
ENTERED PERMANENT WAIT STATE
ENTER 'Y' FOR PROGRAM RESTART
0002FFFF00000000

→ ½

TEST

→ +0=x'ff00000000000800'


TEST

→ go

→ Y


COUNTER:   00011
ENTERED PERMANENT WAIT STATE
ENTER 'Y' FOR PROGRAM RESTART
0002FFFF00000000

→ N
NORMAL SYSTEM TERMINATION
PROGRAM UNDER TEST HAS TERMINATED NORMALLY+
TEST

→ end


READY
```

```
844 1C                  ONE         DC    P'1'
845 010C                TEN         DC    PL2'10'
847 C3D6E4D5E3C5D97A40  TITLE       DC    C'COUNTER: '
850 0100084780000008    CCW         CCW   1,TITLE,X'80',8
858 0100000090000003                CCW   1,0,X'90',3
860 0100083F00000005                CCW   1,NUM,0,5
868 FF02000000000000    WAITPSW     DC    X'FF02000000000000'
870 0002FFFF00000000    STOPPSW     DC    X'0002FFFF00000000'
                                    ORG   X'78'
078 FF00000000000820                DC    X'FF00000000000820'
                                    END
```

EXECUTION SESSION (operator entry shown by →)

```
→   exec ibm370 list

    ALLOC FI(TERM1) DA(*)
    ALLOC FI(TERM2) DA(*)
    ALLOC FI(CARDIN) DA(*)
    ALLOC FI(TAPE1) DA(*)
    ALLOC FI(PRINT) DA(*)
    ALLOC FI(ISTEP1) DA(*)
    ALLOC FI(ISTEP0) DA(*)
    TEST 'CPSC423.PGMLIB.LOAD(IBM370)'
    Q MEMORY
    +800=X'05204130204E50300048F342203D203A96F020419C00000E47702012'
    +81C=X'820206691040044 7E0201AFA20203A2042F921203A204347D02008'
    +838=X'8200206E00000C40404040401C010CC3D6E4D5E3C5D97A4001000847800000008'
    +858=X'0100000090000003010008 3F00000005FF020000000000000002FFFF00000000'
    +78=X'FF00000000000820'
    PSW=X'FF000000000000800'
    TEST

→   list +800 i 1(60)

       +800    BALR  2,0
       +802    LA    3,78(0,2)
       +806    ST    3,72(0,0)
       +80A    UNPK  61(5,2),58(3,2
       +810    OI    65(2),X'F0'
       +814    SIO   14(0)
       +818    BC    7,18(0,2)
       +81C    LPSW  102(2)
       +820    TM    68(0),X'04'
       +824    BC    14,26(0,2)
       +828    AP    58(3,2),66(1,2)
       +92E    CP    58(3,2),67(2,2)
       +834    BC    13,8(0,2)
       +838    LPSW  110(2)
    TEST

→   go

    ENTER IPL ADDRESS OR 'N' FOR NO IPL

→   n

    DO YOU WISH TO ISTEP?   (Y/N)

→   y
```

```
FF00000000000800

FF00000040000802

FF00000080000806

FF00000080000810

FF000000C0000810
→ ½                (break key)
TEST

→ istep=c'N'


TEST

→ go


COUNTER:    00000
COUNTER:    00001
COUNTER:    00002
COUNTER:    00003
COUNTER:    00004
COUNTER:    00005
COUNTER:    00006
COUNTER:    00007
COUNTER:    00008
COUNTER:    00009
COUNTER:    00010
ENTERED PERMANENT WAIT STATE
ENTER 'Y' FOR PROGRAM RESTART
0002FFFF00000000

→ ½

TEST

→ +0=x'ff0000000000080a'


TEST

→ go

→ Y


COUNTER:    00011
ENTERED PERMANENT WAIT STATE
ENTER 'Y' FOR PROGRAM RESTART
0002FFFF00000000

→  N
NORMAL SYSTEM TERMINATION
PROGRAM UNDER TEST HAS TERMINATED NORMALLY+
TEST

→ end


READY
```

The last wait state message is printed when either

1) a PSW is loaded with the WAIT bit set and no interrupts are allowed

or

2) a PSW is loaded with the WAIT bit set and no interrupts are outstanding.

A 'Y' response will load the PSW from location 0 and continue execution.

SAMPLE OPERATING SYSTEM

A small operating system is designed and implemented each semester on the simulator. A typical design usually includes a bootstrap, loader, supervisor, device handlers, and a problem program. The problem program operates in protected memory and requests services from the supervisor through supervisor call interrupts. The supervisor processes all interrupts and coordinates processing among all other modules. The device handlers are responsible for queuing requests for I/O services if devices are busy and starting new I/O operations as devices become available. The problem program is usually a simple application such as a tape print utility using double buffering and channel programming.

CONCLUSION

Algorithms fundamental to the design of operating systems are an important part of every course concerned with operating systems. A simulator for machine hardware provides an excellent tool for reinforcing classroom concepts as students implement and evaluate many of the algorithms described in textbooks. In addition, a study of the simulator design leads to a much better understanding of the functional characteristics of the hardware and its influence on operating systems design.