Computer System Design Description

Yaohan Chu

Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

This paper describes language constructs for creating a very high-level language as a tool for computer system design. In order to describe the complexity of a computer system, this language permits descriptions in a heirarchical manner at different levels of system detail and at various cross-sections of system structure. It allows descriptions of system hardware, system software, and the interactions between system hardware and software.

Introduction

The description of the computer system is of great importance in computer system design. It is important because we need to understand how the hardware works, how the software works, and how the hardware and software interact. We need techniques to describe the computer system for documentation as well as for human communication. The past failure to develop a successful way to describe computer systems physically may be one of the major reasons why computer system design has not been widely recognized as a distinctive and unique engineering subject.

Review of Description Techniques

The modern computer system has been in existence over 30 years, since the advent of the digital computer ENIAC. Techniques have been developed to describe a computer system (system units, system interconnection, system architecture, or system design), yet no technique has satisfactorily described the system at desired levels of details or at chosen cross-sections of system structure. The known techniques include Englsih narrative, flow charts, block diagrams, HIPO charts, and PMS description.

(a) Englsih Narrative

English narrative is the most commonly used technique. It is descriptive and understandable, but it is lengthy and imprecise. It is often used as a supplement to other description techniques.

(b) FLow Chart

The flow chart was first introduced by von Neumann in the early 1950's. It describes a control flow (or a sequence of data operations), but it does not describe the operands or names. It can describe an algorithm at various levels of detail of the computer system, but it does not describe concurrent or parallel operaions. It has been widely used, yet it is highly inadequeate.

(c) Block Diagram

The block diagram is widely used for describing a computer system. It identifies the system units and indicates (partially and incompletely the data and control paths of the computer system. It is pictorial and thus highly descriptive, but it gives no description on actual system architecture and system operations.

(d) HIPO Charts

HIPO, Hierarchy plus Input-Process-Output, consists of two or more component charts. It describes the input, output, and functions of a computer system. The HIPO charts are more useful for analysis or system requirements specifications.

(e) PMS Description

Bell and Newell developed a notation technique for representing computer hardware at the the system level [3]; this technique is known as PMS description where P, M, and S represent processor, memory and switch of a computer system, respectively. (Additionally, there are T for transducer, K for control, D for data operation, L for link, and C for computer.)

The representation of computer hardware at the system level can be done by using these characters and lines to form a diagram called a PMS diagram. However, the PMS diagram fails to describe the lower levels of hardware details. It also fails to describe the sequential operations of the sytem. It describes neither the software part of the computer system, nor the interaction between the hardware and software of the computer system.

0420-0098/82/0000/0842$00.75

Design Descriptio of Computer Systems

Language constructs that can describe a computer system hierarchically are presented below. Language constructs that can describe the hardware, the software, and their interactions are also introduced. The mechanisms that provide interprocessor communication are also presented.

(a) Hierarchical Descriptions

Computer systems involve both hardware and software. Since they can be quite complex, they need to be shown at different levels of system details and at different cross-sections of system structure. Hierarchical description of a computer system can meet such a need.

Hierarchical structure of a computer system can be described by using the level number as a language construct. As an illustration, Fig. 1 shows the hierarchical descriptions (in skeleton) in three levels of computer system XYZ; they are called "level descriptions". These three descriptions called A, B, and C are shown in more details in Figs. 2 to 4. Description A in Fig. 6 identifies system units in four levels in a tree-like relation. Description B in Fig. 3 shows the additional details (the loop statement at level 03 of multiplexor MUX) of the interprocessor communication between the memory on the one hand and the processor and the two channels on the other hand. Description C in Fig. 4 shows the interprocessor control which is achieved by the hardware semaphores in the control loop of each of the four system units. These are the four loop statements at level 03 in each of the memory, processor, and two channels. These descriptions are direct, understandable, concise, and precise. More levels can be used to show additional system details.

Instead of showing more levels, a description can show, for example, the details of channel A; in this case it is called a "cross-section description". By means of the level descriptions and cross section descriptions, a computer system can be described at particular levels of details, and/or at particular cross-sections.

(b) Hardware System Design Description (Fig. 5)

There are a number of computer hardware description languages [1-7] which offer various language constructs to describe computer hardware at different levels of abstraction. CDL [1,4] describes the computer elements at a one-to-one correspondence level including timing and thus has the important advantage of direct visualization of computer hardware. (CDL may also describe asynchronous operation but this is not well known because of the current limitation in the CDL Simulator.)

A study has been undertaken to extend the computer design language, CDL, with additional language constructs to become a microcomputer design language, MDL, [12]. The MDL permits hierarchical description. An example is shown in Fig. 5 which describes a hardware "music playing" microcomputer in 3 levels; level numbers are used to visibly show the hardware structure. The microcomputer consists of a ROM to store the music-score codes and an interpreter (which interprets those codes), an SC/MP microprocessor, and a loudspeaker. The microprocessor reads the music-score code, interprets it, and then generates square waves to drive the loudspeaker at the frequencies and durations specified by the music-score code. Description A is a "chip identification level" as it identifies the chips of the music playing microcomputers: A PROM, an MPU, and a speaker; there are only 2 levels, 01 and 02. Description B is a "chip interconnection level" because it shows the interconnections of all the chips; inner levels, 03, are added to accommodate the interconnection statements. Description C is a "chip description level" which gives the description of the internal structures on the chips and the sequences performed by the chips. Additional levels, 03, are added in the MPU to describe the registers and sequences of the National SC/MP microprocessor except that the registers and the sequences are indicated in a skeleton form for brevity.

(c) Software Design Description (Fig. 6)

It is important and necessary that the hardware and software of a computer system can be treated alike at the system design level. It needs to describe both the software and hardware parts of the computer system, and the hardware and software descriptions should be in a uniform syntax. Chu has extended the CDL to software description and has developed a software design language, SDL-1 [8-10]. An example is shown in Fig. 6 which describes in skeleton a lexical scanner. Level numbers are used to show visibly the program structure, procedure structure [11], data structure, and their reference structures.

The overall consideration about SDL is understandability of the software design. There are two types of statements: control flow statements and data flow statements to describe the control flow and data flow, respectively. SDL is relatively simple in control structure, but rich in data types and data structures. There are adequate data operations provided for these data types and data structures. It is a design language which enables the software engineer to describe a software design which is complete, precise, and structured.

(d) System Hardware/Software Interaction (Fig. 7-10)

System hardware/software interactions means those between system units and the program in execution. Such interactions include the hardware interrupt from a system unit to the central processor as well as from the processor itself, the system call or trap by the program, the operator intervention from the console, the I/O device-enables and data transfers, virtual memory operation, and concurrent and parallel processing among system units. A computer system design language should be capable of describing these interactions

in such a manner that these system functions and operations can be fully understood and their descriptions are precise and concise.

In a conventional computer system, the program being executed is stored in the memory (or partly in the memory and partly on the disk). The approach taken here is to expand the levels of memory description to the level of program details. An example for describing the interaction in skeleton is shown in Fig. 7 where level 03 description of memory indicates the programs (system area, programs, and I/O area).

A more detailed example of the description of a computer system design is shown in Figs. 8 to 10. This example describes the computer system PDP 11/45 at the Department of Computer Science of the University of Maryland. Description A in Fig. 8 identifies system units, system levels, and system bus. As shown, this computer system has two disk units, one Decwriter, one printer, and an interface to the Univac computer systems at the University. Description B in Fig. 9 shows the details of the system program (an input-output system) in the memory. It consists of a memory area for semaphores, one main procedure, the procedure for handling the trap, the procedure for handling the keyboard interrupt, and procedure for handling the typewriter interrupt. The P/V operations for the the two semaphores are incorporated in the trap handler. Descriptions C in Fig. 10 shows further details on hardware and software interaction between the PDP-11 and the system program. It consists of the interrupt handling in processor P, the trap and interrupt vectors, the unibus locations, and the registers for the keyboard and typewriter, and their activations of the interrupt signals. This simple input/output system has actually been implemented on a PDP-11/45 computer system as a student project in a senior operating system course.

The above illustrates a unique approach to describe the interactions between computer system hardware/software at a chosen level and at a chosen cross-section. This approach differs from the traditional one of mapping a software model onto the computer hardware system.

Concluding Remarks

A very high-level language for computer system design can be created with the above constructs to serve as a design tool for describing a computer system design. This language is capable of describing:

(a) a structure of a computer system design at various levels of details and at various cross-sections of structure.
(b) interprocessor communication among processors, memories, I/O interfaces, and other system units,
(c) computer (hardware and software) system architecture, and
(d) the interaction between computer hardware and software.

This language has syntax uniformity in achieving the above capabilities. This language is capable of describing the design directly, understandably, concisely, and precisely. The design described by this language can be used as a high-level "design blueprint" for the computer system design.

References

[1] Chu, Yaohan, "An Algol-like Computer Design Language", Comm. of ACM, October, 1965, pp. 607-615.

[2] Piloty, R., RTS I (Registertransfersprache); 3. Aufl., Institut fur Nachrichtenverarbeitung, TH Darmstadt, 1969.

[3] Bell, G. and Newell, A. Computer Structures: Readings and Examples; McGraw Hill, 1971.

[4] Chu, Yaohan, COMPUTER ORGANIZATION AND MICROPROGRAMMING, Prentice-Hall, Inc., 1972.

[5] Knudsen, M.J., PMSL, An Interactive Language for System Level Description and Analysis of Computer Structures; Ph.D thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Pa. April 1973.

[6] Su, S.Y.H., A Survey of Digital Hardware Descriptive Languages; Proceedings Workshop on Hardware Descriptive Languages. pp. 144, 1974.

[7] Chu, Yaohan, (editor), COMPUTER, Special Issue on "Hardware Description Languages", IEEE Computer Society, December, 1974.

[8] Chu, Yaohan, "Introducing a Software Design Language", Proceedings of the 2nd International Conference on Software Engineering, October, 1976.

[9] Chu, Yaohan, "Introducing Software Blueprint", Proceedings of International Computer Software and Applications Conference, Chicago, 1977.

[10] Chu, Yaohan, "Software Design Language", Technical Report TR-557, Department of Computer Science, University of Maryland, July 1977.

[11] Chu, Yaohan, "Interprocedure Structure", Proceedings of the International Computer Software and Application Conference, Chicago, November 1978.

[12] Chu, Yaohan, "Concepts of a Microcomputer Design Language", Technical Report TR-631, Department of Computer Science, University of Maryland, February 1978.

```
/* HIERARCHICAL DESCRIPTION OF COMPUTER SYSTEM ARCHITECTURE */
/* DESCRIPTION A:  SYSTEM UNITS AND LEVELS */
01  Computer_System XYZ
    02  Memory MAIN
    02  Processor CPU
    02  Channel A
    02  Channel B
    END XYZ
/* DESCRIPTION B:  MEMORY_ACCESS MULTIPLEXING */
01  Computer_System XYZ
    02  Memory MAIN
        03  Bus M(0-31)
            ----------
    02  Processor CPU
        03  Bus P(0-31)
            ----------
    02  Channel A
        03  Bus CHA(0-31)
            ----------
    02  Channel B
            ----------
    END XYZ
/* DESCRIPTION C:  INTERPROCESSOR CONTROL */
01  Computer_System XYZ
    02  Memory MAIN
        03  Bus M(0-31)
            ----------
            LOOP
            ----------
            ENDLOOP
    02  Processor CPU
        03  Bus P(0-31)
            ----------
            LOOP
            ----------
            ENDLOOP
    02  Channel A
        03  Bus CHA(0-31)
            ----------
            LOOP
            ----------
            ENDLOOP
    02  Channel B
        03  Bus CHB(0-31)
            ----------
            LOOP
            ----------
            ENDLOOP
    END XYZ
```

Fig. 1 Hierarchical Description of a computer System Architecture

```
/* HIERARCHICAL DESCRIPTION OF COMPUTER SYSTEM ARCHITECTURE XYZ */
/* DESCRIPTION A:  SYSTEM UNITS AND LEVELS */

01  Computer_System XYZ

    02  Memory MAIN

    02  Processor CPU

    02  Channel A

        03  Controller AX

            04  Keyboard KB

            04  Typewriter TY

        03  Controller AY

            04  Cardreader CR

            04  Printer PTR

    02  Channel B

        03  Controller BX

            04  Disk DK1

            04  Disk DK2

    END XYZ
```

Fig. 2 Hierarchical Description A of a Computer System

```
/* DESCRIPTION B:  MEMORY_ACCESS MULTIPLEXING */
01  Computer_System XYZ
    02  Memory MAIN
        03  Bus M(0-31)
        03  Multiplexor MUX
            LOOP
              WAIT UNTIL IF BUS_REQUEST_CHA THEN M::=::CHA
              ELSE IF BUS_REQUEST_CHB THEN M::=::CHB
              ELSE IF BUS_REQUEST_P THEN M::=::P
              ENDIF ENDIF ENDIF
            ENDLOOP
    02  Processor CPU
        03  Bus P(0-31)
                BUS_REQUEST_P
    02  Channel A
        03  Bus CHA(0-31)
                BUS_REQUEST_CHA
        03  Controller AY
            04  Keyboard KB
            04  Typewriter TY
        03  Controller AY
            04  Cardreader CR
            04  Printer PTR
    02  Channel B
        03  Bus CHB(0-31)
                BUS_REQUEST_CHB
        03  Controller BX
            04  Disk DK1
            04  Disk DK2
    END XYZ
```

Fig. 3 Hierarchical Description B of a Computer system

```
/* DESCRIPTION C:  INTERPROCESSOR CONTROL */
01  Computer_System XYZ
    02  Memory MAIN
        03  Bus M(0-31)
        03  Multiplexor MUX
            LOOP
              WAIT UNTIL IF BUS_REQUEST_CHA THEN CONNECT CHA TO M
              ELSE IF BUS_REQUEST_CHB THEN CONNECT CHB TO M
              ELSE IF BUS_REQUEST_P THEN CONNECT P TO M
              ENDIF ENDIF ENDIF
            ENDLOOP
    02  Processor CPU
        03  Bus P(0-31)
                BUS_REQUEST_P
            Register PC /* Program counter */
            LOOP
              IF INT_REQUEST_CHA THEN PC='memory address for Channel A'
              ELSE IF INT_REQUEST_CHB THEN PC='memory address for Channel B'
              ELSE BEGIN FETCH next-instruction;
                   EXECUTE instruction END ENDIF
              ENDIF
            ENDLOOP
    02  Channel A
        03  Bus CHA(0-31)
                BUS_REQUEST_CHA
                INT_REQUEST_CHA
            LOOP
              WAIT UNTIL RECEIVE I/O instruction;
              EXECUTE_I/O_instruction;
              SEND INTERRUPT_REQUEST_CHA to CPU;
            ENDLOOP
    02  Channel B
        03  Bus CHB(0-31)
                BUS_REQUEST_CHB
                INT_REQUEST_CHB
            LOOP
              WAIT UNTIL RECEIVE I/O instruction;
              EXECUTE_I/O_instruction;
              SEND INTERRUPT_REQUEST_CHB to CPU;
            ENDLOOP
END XYZ
```

Fig. 4 Hierarchical Description C of a Computer System

DESCRIPTION A

```
01  DESIGN SC/MP-MUSIC-BOX
    02  PROM-MM5204 M('address range')
    02  MPU-SC/MP P
    02  SPEAKER SP
    END DESIGN
```

DESCRIPTION B

```
01  DESIGN SC/MP-MUSIC-BOX
    02  PROM-MM5204 M('address range')
        03  PIN
            AD(0-9)::=P.AD(0-9)
            DB(0-7)::=::P.DB(0-7)
            CE::=P.NRDS
    02  MPU-SC/MP P
        03  PIN
            AD(0-11)=::M.AD(0-9)=0
            DB(0-7)::=::M.DB(0-7)
            NWDS
            NRDS=::M.CE
            NADS
            NWOLD::=12
            NRST
            CONT::=12
            ENIN
            ENOUT
            BREQ::=0
            FLAG(0-2)=::SP.SQWAVE
            SENSE(A,B)
            X(1-2)
            S(IN,OUT)
        02  SPEAKER SP
            03  PIN
                SQWAVE::=P.FLAG(0)
        END DESIGN
```

DESCRIPTION C

```
01  DESIGN SC/MP-MUSIC-BOX
    02  PROM-MM5204  M('address range')
        03  PIN
            AD(0-9)::=P.AD(0-9)
            DB(0-7)::=::P.DB(0-7)
            CE::=P.NRDS
    02  MPU-SC/MP P
        03  PIN
            AD(0-11)=::M.AD(0-9)=0
            DB(0-7)::=::M.DB(0-7)
            NWDS
            NRDS=::M.CE
            NADS
            NWOLD::=12
            NRST
            CONT::=12
            ENIN
            ENOUT
            BREQ::=0
            FLAG(0-2)=::SP.SQWAVE
            SENSE(A,B)
            X(1-2)
            S(IN,OUT)
        03  REGISTER
            FR0(0-11)
            FR1(0-11)
            FR2(0-11)
            FR3(0-11)
            A(0-7)
            X(0-7)
            S(0-7)
        03  SEQUENCE FETCH
            ---
            ENDS FETCH
            ---
    02  SPEAKER SP
        03  PIN
            SQWAVE::=P.FLAG(0)
    END DESIGN
```

Fig. 5 Hierarchical Description of a hardware "musical playing" microcomputer

```
01  DESIGN SPECIFICATION
    /*This is the structured design of a lexical scanner*/
    02  PROCEDURES
        03  PROCEDURE DECLARATION
        #1  10 main             /*main program of the scanner*/
        #2     20 scan          /*fetch next symbol and its internal code*/
        #3        30 nextchar   /*returns next character in buffer in*/
        #4           40 lookup  /*search table*/
        #5        30 error      /*prints out error message for illegal char*/
        03  PROCEDURE STRUCTURE
        #1  10 main::=/scan,nextchar
        #2     20 scan::=/nextchar,error,lookup
        #3        30 nextchar::=/lookup
        #4           40 lookup
        #5        30 error
    02  DATA
        03  DATA DECLARATION
        BUFFER
            in OF STRING,     /*holds the source program string*/
            sa OF STRING,     /*assembles characters into a token*/
            i  OF NUMBER,     /*pointer to buffer in*/
            location OF NUMBER;   /*store pointer i for a token*/
        TABLE
            01 token-string,
               02 c OF NUMBER,   /*internal code of each token*/
               02 n OF NUMBER;   /*pointer to BUFFER in*/
        03  REFERENCE STRUCTURE
            BUFFER in::=main,nextchar
                   sa::=scan
                   i::=main,scan,nextchar
                   location::=scan
            TABLE token_string::=main,scan
    02  SWITCHES
        03  SWITCH DECLARATION
            SWITCH class OF STATUS(1,2,3,4,5,6);
        03  SWITCH STRUCTURE
            class::=nextchar/scan
    02  DEFINITIONS
        [definitions of all procedures]
END_DESIGN
```

Fig. 6 A Software Description in Software Design Language SDL-1

```
01  Computer_System XYZ

    02  Memory MAIN
        03  Bus M(0-31)
        03  Multiplexor MUX
            LOOP
              WAIT UNTIL IF BUS_REQUEST_CHA THEN CONNECT CHA TO M
              ELSE IF BUS_REQUEST_CHB THEN CONNECT CHB TO M
              ELSE IF BUS_REQUEST_P THEN CONNECT P TO M
              ENDIF ENDIF ENDIF
            ENDLOOP
        03  Location (0-1K)

            ---description of system area---

        03  Location (1K-60K)

            ---description of programs---

        03  Location (60-64K)

            ---description of I/O area---

    02  Processor CPU

    02  Channel A

    02  Channel B
        03  Bus CHB(0-31)
                BUS_REQUEST_CHB
                INT_REQUEST_CHB
            LOOP
              WAIT UNTIL RECEIVE I/O instruction;
              EXECUTE_I/O_instruction;
              SEND INTERRUPT_REQUEST_CHB to CPU;
            ENDLOOP
        03  Controller RK
            04  Disk DK1  /* system disk */

            ---description of system programs---

            04  Disk DK2  /* user disk */

    END XYZ
```

Fig. 7 Hierarchical Description of Fig. 8 now showing where the software descriptions are located.

```
/* Hierarchical Description of UOM PDP 11/45 System */

/* Description A:  System levels, units, and bus */

01 Computer_System PDP11/45

   02 Memory M(0-56,000)

      03 Program

   02 Processor P

   02 Bus U(0-55)  /* unibus */

      A(17-00),  /* address bus */
      D(15-00),  /* data bus */
      C(1-0),    /* control bus */
      MSYN,      /* master sync */
      SSYN,      /* slave sync */
      PA,        /* parity available */
      PB,        /* parity bit */
      NPR        /* non-processor request */
      BR(7-4)    /* bus request */
      NPG        /* non-processor grant */
      BG(7-4)    /* bus grant */
      SACK       /* selection acknowledge */
      BBSY       /* bus busy */
      INTR       /* interrupt */
      INIT       /* initialize */
      ACLO       /* AC line low */
      SP         /* spare */

   02 Interface DC11#1

      03  Printer P300

   02 Interface DC11#0

      03 UNIVAC 1108

      03 UNIVAC 1100/42

   02 DMA  DK

      03 Disk DK0

      03 Disk DK1

   02 DECwriter DEC

      03 Keyboard KB

      03 Typewriter TT

   END PDP11/45
```

Fig. 8 Description A of Computer System PDP 11 at UOM

```
/* HIERARCHICAL DESCRIPTION OF COMPUTER SYSTEM PDP11 */
/* DESCRIPTION B:  PROGRAM DESCRIPTION */
01 COMPUTER_SYSTEM PDP11

   02 Processor P      /* CPU of the PDP11/45 */

   02 Bus U(0-55)      /* Unibus of the system */

   02 Memory M(o-56000)        /* Main Memory */

      03 Program

         Semaphore RSEM,                  /*keyboard busybit   */
                   PSEM;                  /*typewriter busybit */

         Procedure MAIN

            BUS_LOC(000032-000033):=000340;  /*Insert Program status */
            BUS_LOC(000062-000063):=000200;  /*Insert Program status */
            Set PSEM to 1;                   /*Initialize semaphore PSEM */
            Set RSEM to 0;                   /*Initialize semaphore RSEM */

            Loop
              Set KYSR(7,0) to 1;            /*enable keyboard and interrupt bits */
              Call EMT_HANDLER(PRD);
              Call EMT_HANDLER(PPR);
              TYDB:=KYDB;                    /*transfer to typewriter data buffer */
              Set TYSR(7) to 1;              /*enable typewriter-interrupt bit */
            Endloop

         END MAIN


         Interrupt _Procedure KB_HANDLER;

            Set KYSR(6-0) To 100Q;
            Call EMT_HANDLER(VRD);

            Interrupt_Return;
         End KB_HANDLER
         Interrupt_Procedure TY_HANDLER;

            Set TYSR(6-0) To 100Q;
            Call EMT_HANDLER(VPR);

            Interrupt_Return;
         End TY_HANDLER
```

Fig. 9 Description B of Computer System PDP 11 at UOM

```
         Interrupt_Procedure EMT_HANDLER(FY);

            Change program-status;
            Save registers;

            Case FV Of

            PRD:   /*P operation for keyboard */
                   Decrement PSEM by 1;
                   Loop
                     Wait Until PSEM = 0
                   Endloop;

            VRD:   /*V operation for keyboard */
                   Increment RSEM by 1;

            PPR:   /*P operation for typewriter */
                   Decrement RSEM by 1;
                   Loop Wait Until RSEM=0 Endloop;

            VPR:   /*V operation for typewriter */
                   Increment PSEM by 1;

            Endcase

            Restore registers;

            Interrupt_Return;

         End EMT_HANDLER

      02 Interface DC11#1     /* Asynchronous Serial Interface */

      02 Interface DC11#0     /* Asynchronous Serial Interface */

      02 Direct-Memory_Access DK

      02 Decwriter DWR

         03 Keyboard KB
         03 Typewriter TY

      END PRP11
```

Fig. 9 Continued

```
/* HIERARCHICAL DESCRIPTION OF COMPUTER SYSTEM PDP11 */
/* DESCRIPTION C:  HARDWARE AND SOFTWARE INTERACTION */
01 COMPUTER_SYSTEM PDP11

   02 Processor P      /* CPU of the PDP11/45 */

      Loop
        If INTR_KB Then Transfer-To KB_HANDLER  Endif;
        If INTR_TT Then Transfer-To TT_HANDLER  Endif;
        Fetch Next Instruction;
        If EMT-Instruction Then Transfer-To EMT_HANDLER
           Else Execute Instruction Endif;
      Endloop;

   02 Bus U(0 -55)     /* Unibus of the system */

      03 Address_Bus A(17-0)    /* bus locations on address bus */

         BUS_LOC(777562-777563)=KYDB(15-0)      /*keyboard data buffer register */
         BUS_LOC(777560-777561)=KYSR(15-0)      /*keyboard status register */
         BUS_LOC(777566-777567)=TTDB(15-0)      /*typewriter data buffer register */
         BUS_LOC(777564-777565)=TTSR(15-0)      /*typewriter status register */
         BUS_LOC(177776-177777)=PS(15-0)        /*processor status word */
         BUS_LOC(000030-000033)=EMT_HANDLER     /*location of EMT-handler subroutine*/
         BUS_LOC(000060-000063)=KB_HANDLER      /*location of keyboard handler subr.*/
         BUS_LOC(000064-000067)=TT_HANDLER      /*location of typewriter handler subr.*/

   02 Memory M(o-56000)          /* Main Memory */

      03 Program

         Semaphore RSEM,                  /*keyboard busybit   */
                   PSEM;                  /*typewriter busybit */

         Procedure MAIN

            BUS_LOC(000032-000033):=000340;     /*Insert Program status */
            BUS_LOC(000062-000063):=000200;     /*Insert Program status */
            Set PSEM to 1;                      /*Initialize semaphore PSEM */
            Set RSEM to 0;                      /*Initialize semaphore RSEM */

            Loop
              Set KYSR(7,0) to 1;               /*enable keyboard and interrupt bits */
              Call EMT_HANDLER(PRD);
              Call EMT_HANDLER(PPR);
              TTDB:=KYDB;                       /*transfer to typewriter data buffer */
              Set TTSR(7) to 1;                 /*enable typewriter-interrupt bit */
            Endloop.

         END MAIN
```

Fig. 10 Description C of Computer System PDP 11 at UOM

```
         Interrupt _Procedure KB_HANDLER:

            Set KYSR(6-0) To 1000;
            Call EMT_HANDLER(VRD);

            Interrupt_Return;
         End KB_HANDLER
         Interrupt_Procedure TT_HANDLER:

            Set TTSR(6-0) To 1000;
            Call EMT_HANDLER(VPR);

            Interrupt_Return;
         End TT_HANDLER
         Interrupt_Procedure EMT_HANDLER(PT);

            Change program-status;
            Save registers;

            Case PV Of

            PRD:   /*P operation for keyboard */
                   Decrement PSEM by 1;
                   Loop
                     Wait Until PSEM = 0
                   Endloop;

            VRD:   /*V operation for keyboard */
                   Increment RSEM by 1;

            PPR:   /*P operation for typewriter */
                   Decrement RSEM by 1;
                   Loop Wait Until RSEM=0 Endloop;

            VPR:   /*V operation for typewriter */
                   Increment PSEM by 1;

            Endcase

            Restore registers;

            Interrupt_Return;
          End EMT_HANDLER
      Endprogram

   02 Interface DC11#1     /* Asynchronous Serial Interface */

   02 Interface DC11#0     /* Asynchronous Serial Interface */

   02 Direct-Memory_Access DK

   02 Decwriter DWR

      03 Keyboard KB

         04 Bus INTR_KB             /*interrupt request from keyboard */
         04 Register KYSR(15-0)     /*keyboard status register */
         04 Register KYDB(15-0)     /*keyboard data buffer register */

            When completion then set INTR_KB;

      03 Typewriter TT

         04 Bus INTR_KB             /*interrupt report form typewriter */
         04 Register TTSR(15-0)     /*typewriter status register */
         04 Register TTDB(15-0)     /*typewriter data buffer register */

            When completion then set INTR_TT;

   END PDP11
```

Fig. 10 Continued