|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | USN |  |  |  |  |  |  |  |  |  |  | | | | | | |  | | | | |
| Internal Assessment Test 1 Answer Key – Mar. 2018 | | | | | | | | | | |
| Sub: | System Software | | | | | Sub Code: | 17MCA25 | Branch: | MCA | |
| Date: | 14/03/2018 | Duration: | 90 min’s | Max Marks: | 50 | Sem | II | | | OBE |

**1 (a) Define system software? [02]**

System Software consists of a variety of programs that support the operation of a computer. It makes possible for the user to focus on an application or other problem to be solved, without needing to know the details of how the machine works internally.

They are usually related to the architecture of the machine on which they are to run.

Example: Assembler, Compiler, text editor, loader and linkers etc.

**1 (b) Write general description of pass 1 and pass 2 for two pass assembler[08]**

Pass 1 (define symbols)

* Assign addresses to all statements in the program
* Save the addresses assigned to all labels for use in Pass 2
* Perform some processing of assembler directives, (including those for address assignment, such as BYTE and RESW

Pass 2 (assemble instructions and generate object program)

* Assemble instructions (translate opcodes and look up addresses)
* Generate data values defined by BYTE, WORD etc.
* Perform processing of assembler directives not done during Pass 1
* Write the object program and the assembly listing

**2 (a) Differentiate between application software and system software. [04]**

|  |  |
| --- | --- |
| System Software | Application Software |
| Intended to support the operation and use of the computer | An application program is primarily concerned with the solution of some problem, using the computer as tool |
| Focus is on the Computer system and not on the application | The focus is on the application not on the computing system. |
| It depends on the structure of the machine on which it is executed. | It does not depend on the structure of the machine it works |
| Ex. Operating system, Loader, Linkers, assembler, compiler, text editors etc. | Ex. Banking system, Inventory system. |

**2 (b) Write Simple object program format. (Header, Text, End Record) [06]**

The simple object program format contains three types of records: Header record,

Text record and end record. The header record contains the starting address and length. Text record contains the translated instructions and data of the program, together with an indication of the addresses where these are to be loaded.

The end record marks the end of the object program and specifies the address

where the execution is to begin.

The format of each record is as given below.

Header record:

Col 1 H

Col. 2-7 Program name

Col 8-13 Starting address of object program (hexadecimal)

Col 14-19 Length of object program in bytes (hexadecimal)

Text record:

Col. 1 T

Col 2-7. Starting address for object code in this record (hexadecimal)

Col 8-9 Length off object code in this record in bytes (hexadecimal)

Col 10-69 Object code, represented in hexadecimal (2 columns per byte of object

code)

End record:

Col. 1 E

Col 2-7 Address of first executable instruction in object program

(hexadecimal)

**3 Describe SIC (standard model) Machine Architecture. [10]**

1) Memory

* Memory consists of 8-bit bytes.
* 3 consecutive bytes form a word (24 bits).
* All the address in SIC are byte addresses.
* Words are addressed by the location of their lowest numbered byte.
* There are total of 32,768 (215) bytes in the computer memory.

2) Registers

|  |  |  |
| --- | --- | --- |
| There are five registers, each 24 bits in length. Mnemonic | Number | Use |
| A | 0 | Accumulator; used for arithmetic operations |
| X | 1 | Index register; used for addressing |
| L | 2 | Linkage register, the jump to subroutine instruction stores the return address in this register. |
| PC | 8 | Program counter, contains the address of the next instruction to be fetched for execution. |
| SW | 9 | Status word, contains a variety of information, including a Condition Code. |

3) Data Formats

* Integers are stored as 24 bit binary numbers; 2‟s complement representation is used for negative values.
* Characters are stored using their 8-bit ASCII codes.
* There is no floating point hardware on the standard version of SIC.

4) Instruction Formats

All machine instructions on the standard version of SIC have the following 24-bit format 8 1 15

|  |  |  |
| --- | --- | --- |
| opcode | x | address |

8 1 15

The flag bit x is used to indicate indexed addressing mode.

5) Addressing Modes

There are two addressing modes, indicated by the setting of the x bit in the instruction.

|  |  |  |
| --- | --- | --- |
| **Mode** | **Indication** | **Target address calculation** |
| Direct | x = 0 | TA = address |
| Indexed | x = 1 | TA = address + (x) |

Parentheses are used to indicate the contents of a register or a memory location. For example, ( X ) represents the contents of register X.

6) Instruction Set

SIC provides a basic set of instructions that are sufficient for most simple task.

i) Data transfer instruction: This include instructions that load and store registers. Eg. LDA, LDX, STA, STX.

ii) Arithmetic operation instruction: Basic arithmetic operations that involves register A Eg. ADD, SUB, MUL, DIV, COMP.

iii) Conditional Branching: Conditional jump instructions test the settings of conditional code and jump accordingly. Eg. JLT, JGT, JEQ.

iv) Subroutine call Instructions: Perform subroutine linkage. Eg. JSUB, RSUB. Return address is stored in linkage(L) register.

7) Input and Output

 Input and Output are performed by transferring 1 byte at a time to or from the rightmost 8 bits of register A (accumulator).

 Each device is assigned a unique 8bit code.

 There are 3 I/O instructions.

 The Test Device (TD) instruction tests whether the addressed device is ready to send or receive a byte of data. Read Data (RD), Write Data (WD) are used for reading or writing the data.

**4 Describe SIC/XE Machine Architecture. [10]**

1) Memory

* Memory consists of 8-bit bytes.
* 3 consecutive bytes form a word (24 bits).
* All the address in SIC/XE are byte addresses.
* Words are addressed by the location of their lowest numbered byte.
* Maximum memory available on a SIC/XE system is 1 megabyte (220 bytes).
* This increase leads to a change in instruction formats and addressing modes

2) Registers

Five registers of SIC machine remains same in SIC/XE. The additional registers provided by SIC/XE are as follows.

|  |  |  |
| --- | --- | --- |
| Mnemonic | Number | Use |
| B | 3 | Base register; used for addressing. |
| S | 4 | General working register – no special use. |
| T | 5 | General working register – no special use. |
| F | 6 | Floating-point accumulator (48 bits). |

3) Instruction Formats

* SIC/XE has larger memory hence instruction format of standard SIC version is no longer suitable.
* SIC/XE provide two possible options; using relative addressing (Format 3) and extend the address field to 20 bit (Format 4).
* In addition SIC/XE provides some instructions that do not reference memory at all. (Format 1 and Format 2) .
* The new set of instruction format is as follows. Flag bit e is used to distinguish between format 3 and format 4. (e=0 means format 3, e=1 means format 4)

1. Format 1 (1 byte)

8

|  |
| --- |
| op |

Example RSUB (return to subroutine)

opcode

|  |
| --- |
| 0100 1100 |

4 C

1. Format 2 (2 bytes)

8 4 4

|  |  |  |
| --- | --- | --- |
| op | r1 | r2 |

Example COMPR A, S (Compare the contents of register A & S)

Opcode A S

|  |  |  |
| --- | --- | --- |
| 1010 0000 | 0000 | 0100 |

A 0 0 4

1. Format 3 (3 bytes)

6 1 1 1 1 1 1 12

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| op | n | i | x | b | p | e | disp |

Example LDA #3(Load 3 to Accumlator A)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0000 00 | 0 | 1 | 0 | 0 | 0 | 0 | 0000 0000 0011 |

0 n i x b p e 0 0 3

1. Format 4 (4 bytes)

6 1 1 1 1 1 1 20

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| op | n | i | x | b | p | e | address |

Example JSUB RDREC(Jump to the address, 1036)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0100 10 | 1 | 1 | 0 | 0 | 0 | 1 | 0000 0001 0000 0011 0110 |

n i x b p e

4) Addressing Modes

Two new relative addressing modes are available for use with instructions assembled using Format 3

|  |  |  |
| --- | --- | --- |
| **Mode** | **Indication** | **Target address calculation** |
| Base Relative | b=1, p=0 | TA = (B) + disp ( 0≤ disp ≤ 4095) |
| Program-counter relative | b=0, p=1 | TA = (PC)+disp (-2048 ≤ disp ≤ 2047) |

b represents for base relative addressing where as p represents program counter relative addressing. If both the bits b and p are 0 then target address is taken form the address field of the instruction (i.e displacement)

SIC/XE also support addressing modes that are assembled using Format 4.

|  |  |  |
| --- | --- | --- |
| **Mode** | **Indication** | **Target address calculation** |
| Direct | b=0, p=0, x=0 | TA = disp |
| Indexed | x=1 | TA = (x)+disp |
| Immediate | i=1, n=0 | TA = operand value |
| Indirect | i=0, n=1 | TA = address of operand value |
| simple | i=1, n=1 i=0, n=0 | TA = location of the operand value |

6) Instruction Set

* SIC/XE provides all of the instructions that are available on the standard version.
* In addition we have, Instructions to load and store the new registers LDB, STB, etc,
* Floating-point arithmetic operations, ADDF, SUBF, MULF, DIVF,
* Register move instruction : RMO,
* Register-to-register arithmetic operations, ADDR, SUBR, MULR, DIVR and,
* Supervisor call instruction : SVC

7) Input and Output

* There are I/O channels that can be used to perform input and output while the CPU is executing other instructions.
* Allows overlap of computing and I/O, resulting in more efficient system operation.
* The instructions SIO, TIO, and HIO are used to start, test and halt the operation of I/O channels.

**5 Write pass 1 Algorithm for pass two assembler [10]**

****

**6 Write pass 2 Algorithm for pass two assembler[10]**



**7 Give the target address generated for following machine instruction. [10]**

**03C300 h if (B)=006000**

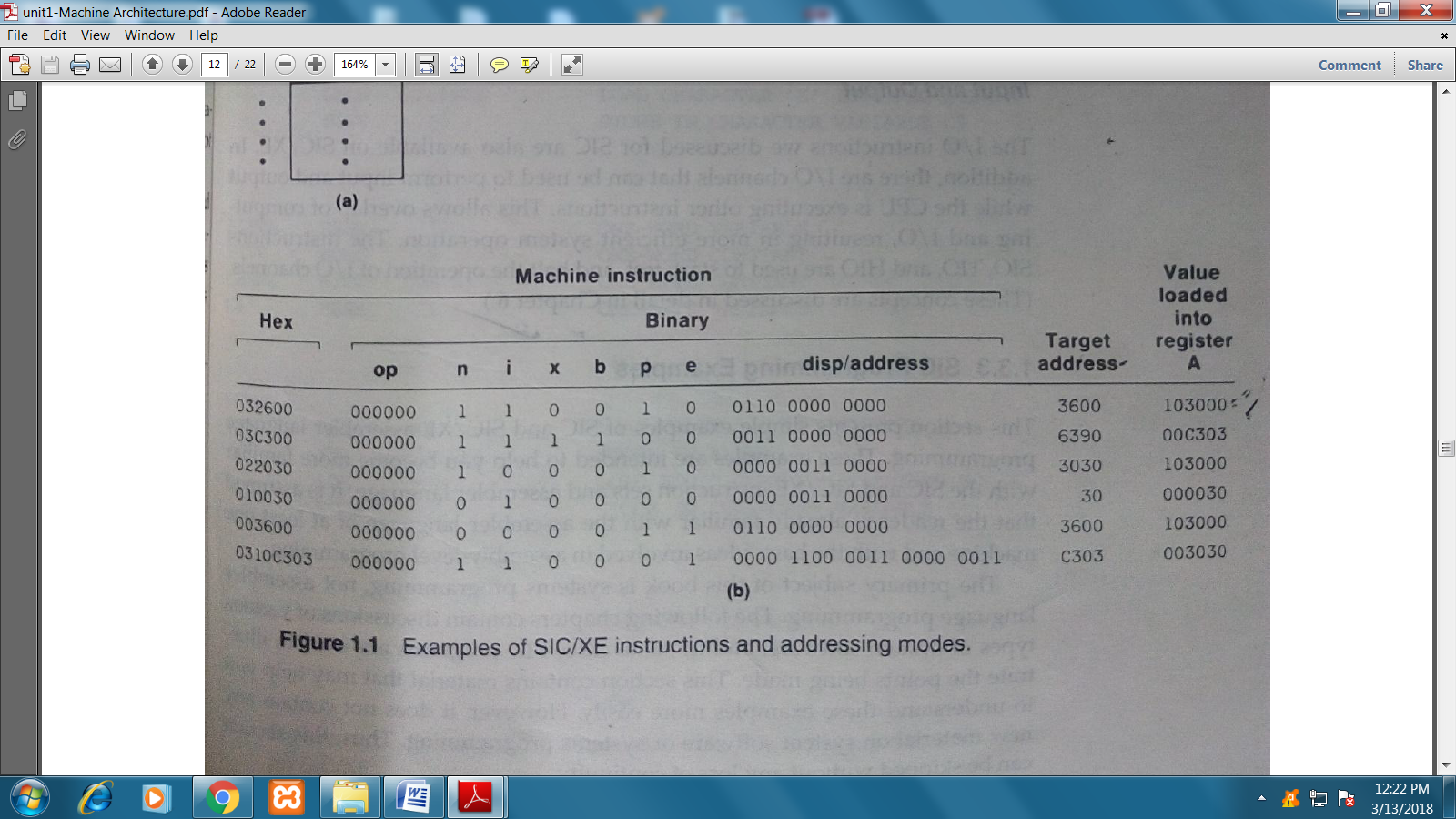
**010030 h (PC)=003000**

**022030 h (x)=000090**

**003600 h**

**032600 h**

**0310C303 h**



**8 a) Write a program for SIC/XE machine to copy a string “Master of Computer Applications” from LOC1 to LOC2 [5]**

LDT #31

LDX #0

MOVECH LDCH LOC1,X

STCH LOC2,X

TIXR T

JLT MOVECH

.

.

LOC1 BYTE C’ Master of Computer Applications’

LOC2 RESB 31

**8 b) Write a assembly language program in SIC/XE to add 2 arrays of 200 integers**

LDS #3

LDT #600

LDX #0

ADDLP LDA ALPHA,X

ADS BETA,X

STA GAMMA,X

ADDR S,X

COMPR X,T

JLT ADDLP

ALPHA RESW 200

BETA RESW 200

GAMMA RESW 200

**9 a) List and explain any five assembler directives with examples. [5]**

**Assembler Directives**

In addition to the mnemonic machine instructions assembler uses following

assembler directives. These statements are not translated into machine instructions.

Instead they provide instructions to assembler itself.

1) START

START specify the name and starting address of the program.

Example: START 1000

2) END

Indicate the end of the source program and (optionally) specify the first

executable instruction in the program.

Example: END FIRST

3) BYTE

Generate character or hexadecimal constant, occupying as many bytes as

needed to represent the constant.

Example: BYTE X’F1’

4) WORD

Generate one-word integer constant

Example: THREE WORD 3

5) RESB

Reserve the indicate number of bytes for a data area.

Example: BUFFER RESB 4096

6) RESW

Reserve the indicate number of words for a data area.

Example: LENGTH RESW 1

**9 b) List and describe data structures used by two-pass assembler [5]**

**1) OPTAB:**

 It is used to lookup mnemonic operation codes and translates them to

their machine language equivalents.

 In more complex assemblers the table also contains information about

instruction format and length

 In pass 1 the OPTAB is used to look up and validate the operation

code in the source program.

 In pass 2, it is used to translate the operation codes to machine

language.

 In simple SIC machine this process can be performed in either in pass

1 or in pass 2.

 But for machine like SIC/XE that has instructions of different lengths,

we must search OPTAB in the first pass to find the instruction length

for incrementing LOCCTR.

 In pass 2 we take the information from OPTAB to tell us which

instruction format to use in assembling the instruction, and any

peculiarities of the object code instruction.

 OPTAB is usually organized as a hash table, with mnemonic

operation code as the key.

 The hash table organization is particularly appropriate, since it

provides fast retrieval with a minimum of searching.

 Most of the cases the OPTAB is a static table- that is, entries are not

normally added to or deleted from it. In such cases it is possible to

design a special hashing function or other data structure to give

optimum performance for the particular set of keys being stored.

2) **SYMTAB:**

 This table includes the name and value for each label in the source

program, together with flags to indicate the error conditions (e.g., if a

symbol is defined in two different places).

 During Pass 1: labels are entered into the symbol table along with

their assigned address value as they are encountered. All the symbols

address value should get resolved at the pass 1.

 During Pass 2: Symbols used as operands are looked up the symbol

table to obtain the address value to be inserted in the assembled

instructions.

 SYMTAB is usually organized as a hash table for efficiency of

insertion and retrieval. Since entries are rarely deleted, efficiency of

deletion is the important criteria for optimization.

3) **LOCCTR:**

 Apart from the SYMTAB and OPTAB, this is another important

variable which helps in the assignment of the addresses.

 LOCCTR is initialized to the beginning address mentioned in the

START statement of the program.

 After each statement is processed, the length of the assembled

instruction is added to the LOCCTR to make it point to the next

instruction.

 Whenever a label is encountered in an instruction the LOCCTR value

gives the address to be associated with that label.

**10a) Write a short note on UltraSPARC Architecture [5]**

**UltraSPARCArchitecture**

UltraSPARC processor, announced by Sun Microsystems in 1995, is the last member of the SPARC family.

**Memory**

 Memory consist of 8-bit bytes

 2 consecutive bytes form half word, 4 consecutive bytes form word, 8 consecutive bytes form doubleword.

 UltraSPARC programs can be written using a virtual address space of 264 bytes.

**Registers**

 The SPARC architecture includes a large register file that contains more than 100 general-purpose registers.

 Exact number of register varies from one implantation to another.

 However, any procedure can access only 32 registers designated r0 through r31.

 Registers r0 to r7 are global- they can be accessed by all the procedures on the system.

 Other 24 registers can be visualized as a window through which part of the register file can be seen.

**Data Formats**

 Integers are stored as 8-,16-,32-, or 64-bit binary numbers.

 2’s compliment representation is used for negative values.

 Characters are stored using their 8-bit ASCII codes.

 There are three different floating point data formats.

**Instruction Format**

 There are 3 basic instruction formats in the SPARC architecture.

 All the formats are 32 bit long.

 First 2 bits of the instruction identify which format is being used.

 Format 1 is used for call instruction

 Format 2 is used for branch instructions.

 Format 3 is used for register load and store and arithmetic operations.

|  |  |
| --- | --- |
| **Addressing mode** Mode | Target address calculation |
| Pc-relative | TA=(PC) + displacement |
| Register indirect with displacement | TA=(register)+ displacement |
| Register indirect indexed | TA=(register-1)+ (register-2) |

**10 b) Write a short note on VAX Architecture [5]**

**VAX Architecture**

VAX family ofcomputers was introduce by Digital equipment corporation (DEC) in 1978. VAXarchitecture was designed for the compatibility with the earlier PDP-11 machine. A compatibility mode was provided at hardware level so PDP-11 programs could run unchanged on VAX.

**Memory**

 The VAX memory consist of 8-bit bytes

 2 consecutive bytes form word, 4 consecutive bytes form long word, 8 consecutive bytes form quadward, and 16 consecutive bytes form an octaword.

 All VAX programs operate in a virtual address space of 232 bytes.

 One half of the VAX virtual address space is called ystem space which contains operating system and is shared by all the programs.

 The other half of the address space is called process space and and is defined separately for each program.

**Registers**

 There are are 16 general purpose registers on the Vax, denoted by Ro to R15, all are 32 bits in length.

 R15 is program counter, R14 is stack pointer, R13 is frame pointer, R12 is argument pointer, R11to R6 have no special functions and R0 to R5 are available for general use.

**Data Formats**

 Integers are stored as binary numbers in byte, word, longword, quadword or octaword.

 2’s compliment representation is used for negative values.

 Characters are stored using their 8-bit ASCII codes.

 There are four different floating point data formats on the VAX, ranging in length from 4 to 16 bytes.

**Instruction Format**

 Vax machineinstruction use a variable- length instruction format.

 Each instruction consist of an operation code(1 or 2 bytes) followed by up to six operand specifiers, depending on the type of instruction.

 Each operand specifier designates one of the VAX addressing modes and gives any additional information necessary to locate the operand.

**Addressing mode**

VAX provide large number of addressing modes.

 register mode

 register deferred mode

 autoincrement and autodecrement modes

 several base relative addressing modes

 program-counter relative modes

 indirect addressing mode (called deferred modes)

 immediate operands

**Instruction Set**

Goal of the VAX designers was to produce an instruction set that is symmetric with respect to data type

The instruction mnemonics are formed by

 a prefix that specifies the type of operation

 a suffix that specifies the data type of the operands

 a modifier that gives the number of operands involved

**Input and Output**

 Input and output on the VAX are accomplished by I/O device controllers

 Each controller has a set of control/status and data registers, which are assigned locations in the physical address space (called *I/O space*)

 No special instructions are required to access registers in I/O space

 The association of an address in I/O space with a physical register in a device controller is handled by the memory management routines