MODULE 1: SOFTWARE AND HARDWARE

Software
- Definition: A set of instructions given to the computer.
- Characteristics:
  - Intangible (cannot be touched or felt).
  - Developed by writing instructions in a programming language.
  - Controls the operations of the computer.
  - Can be reinstalled from a backup copy if damaged or corrupted.
- Examples: Antivirus, Microsoft Office Tools.

Hardware
- Definition: Physical parts of a computer.
- Characteristics:
  - Tangible (can be touched and felt).
  - Constructed using physical components.
  - Operates under the control of software.
  - Can be replaced if damaged.
- Examples: Keyboard, Monitor, Mouse.

Software vs. Hardware
Software	**Hardware**
Collection of instructions that tells the computer what to do	Physical elements of a computer
Divided into:	Categories:
a. System Software	a. Input Devices
b. Application Software	b. Output Devices
c. Utility Software	c. Storage Devices
Must be installed on the computer	Usable once software is loaded
Prone to viruses	Not susceptible to virus attacks
If damaged/corrupted, reinstallation is possible	If damaged, can be replaced
Examples: Microsoft Office, Adobe	Examples: Mouse, Monitor, Keyboard

Types of Software

1. System Software
- Definition: A collection of programs that support the operation of a computer.
- Functions:
  - Helps run computer hardware and the computer system.
  - Manages the operation of computer hardware.
- Types:
  a. **Operating System**
  b. **Language Translators**
    - i. Compiler
    - ii. Assembler
    - iii. Interpreter
    - iv. Macro Processor

c. **Loader**
d. **Linker**
e. **Debugger**
f. **Text Editor**

2. Application Software
- Definition: Software that allows end users to accomplish one or more specific tasks.
- Focus: Solving specific applications or problems.

Operating System
- Definition: Acts as an interface between the user and the system.
- Features:
  - Provides a user-friendly interface.
- Functions:
  a. Process Management
  b. Memory Management
  c. Resource Management
  d. I/O Operations
  e. Data Management
  f. Provides security for jobs.

Language Translators
- Definition: Programs that convert code from one programming language to another (e.g., source code to object code).
- Types:
  1. **Compilers**
     - Translates a high-level language program into machine language as a whole.
     - Used by: C, C++.
     - Process: Syntax analysis, semantic analysis, and intermediate code generation.
  2. **Interpreters**
     - Translates a high-level language program into machine language line by line.
     - Used by: Ruby, Perl, Python, PHP.
     - Interpretation Cycle:
       i. Fetch the statement.
       ii. Analyze the statement and determine its meaning.
       iii. Execute the meaning of the statement.
  3. **Assemblers**
     - Translates assembly language (mnemonic symbols) into machine language.
     - Assigns machine addresses to a symbol table.

Compiler vs. Interpreter vs. Assembler
Compiler	**Interpreter**	**Assembler**
Converts high-level language to machine language	Translates high-level language to machine language	Converts assembly language to machine language

| Processes the entire program at once | Processes the program line by line | Converts assembly code to machine code |
| Used by: C, C++ | Used by: Ruby, Perl, Python, PHP | Used by: Assembly language |

Linker
- Definition: A process that collects and combines various pieces of code and data into a single executable file.
- Types:
  a. **Linking Loader**: Performs linking and relocation directly into main memory for execution.
  b. **Linkage Editor**: Produces a linked version of the program (load module or executable image) for later execution.
  c. **Dynamic Linker**: Postpones linking until execution time (dynamic loading).

Loader
- Definition: A utility of the operating system that copies a program from a storage device to the computer's main memory.
- Features:
  - Replaces virtual addresses with real addresses.
  - Invisible to the user.

Debugger
- Definition: An interactive system that helps programmers test and debug programs.
- Functions:
  - Locates bugs or faults in the program.
  - Assists in fixing errors.
  - Determines the exact nature and location of errors.

Device Driver
- Definition: A software module that manages communication and control of specific I/O devices.
- Function: Converts logical requests from the user into specific commands for the device.

Macro Processor
- Definition: A program that processes macros, which are code fragments defined once and reused by calling them.
- Features:
  - Defined using the `#define` directive (e.g., `#define BUFFER_SIZE 1020`).
  - Embedded in assemblers and compilers.

Text Editors
- Definition: Programs that allow users to create and edit source programs as text in main memory.
- Functions: Creation, editing, deletion, and updating of documents or files.

Simplified Instructional Computer (SIC)
- Definition: A hypothetical computer with hardware features found in real machines.

- Versions:
  a. SIC Standard Model
  b. SIC/XE (Extra Equipment)

Machine-Dependent Features of Software Systems
1. **Assembler**: Instruction format, addressing mode.
2. **Compiler**: Registers, machine instructions.
3. **Operating System**: All resources of the computing system.

Machine-Independent Features of Software Systems
1. General design and logic of the assembler.
2. Code optimization in the compiler.
3. Linking independently assembled subprograms.

SIC Architecture - Standard Model
1. **Memory**: 2^{15} bytes (32,768 bytes).
2. **Registers**:
   - Five registers, each 24 bits long.
Mnemonic	Number	Special Use
A	0	Accumulator; used for arithmetic operations
X	1	Index register; used for addressing
L	2	Linkage register; stores return address for JSUB
PC	8	Program counter; address of next instruction
SW	9	Status word; contains condition code (CC)
3. **Data Formats**:
   - Integers: 24-bit binary numbers (2's complement for negatives).
   - Characters: 8-bit ASCII codes.
   - No hardware support for floating-point numbers.
4. **Instruction Format**:
   - 24-bit format: | 8 bits (OPCODE) | 1 bit (X) | 15 bits (Address) |
   - Flag bit `X` indicates indexed addressing mode.
5. **Addressing Modes**:
   a. **Direct Addressing Mode**: `X = 0`, Target Address = Actual Address.
   b. **Indexed Addressing Mode**: `X = 1`, Target Address = Address + (X).
6. **Instruction Set**:
   a. **Data Transfer**: LDA, STA, LDX, STX.
   b. **Arithmetic Operations**: ADD, SUB, MUL, DIV, COMPR.
   c. **Conditional Branching**: JLT, JEQ, JGT.
   d. **Subroutine Call**: JSUB (jump), RSUB (return).
   e. **Input/Output**: TD (test device), RD (read data), WD (write data).
7. **Input and Output**:
   - Transfers 1 byte at a time to/from the rightmost 8 bits of register A.
   - Each device has a unique 8-bit code.
8. **Data Movement and Storage Definitions**:
   - LDA, STA, LDX, STX: 3-byte words.
   - LDCH, STCH: 1-byte characters.
   - Storage Definitions:

- WORD: One-word constant.
- RESW: One-word variable.
- BYTE: One-byte constant.
- RESB: One-byte variable.

SIC/XE Architecture - SIC with Extra Equipment

1. **Memory**: Maximum 1 MB (2^{21} bytes).
2. **Registers**:
   - Additional registers: B, S, T, F (in addition to SIC registers).
Mnemonic	Number	Special Use
B	3	Base register
S	4	General working register
T	5	General working register
F	6	Floating-point accumulator (48 bits)
3. **Floating-Point Data Type**:
   - 48-bit format: | 1 bit (s) | 11 bits (exponent) | 36 bits (fraction) |
   - Formula: F * 2^(e - 1024).
4. **Instruction Formats**:
   a. **Format 1 (1 byte)**: | 8 bits (op) |
      - Example: RSUB (Return to Subroutine).
   b. **Format 2 (2 bytes)**: | 8 bits (op) | 4 bits (r1) | 4 bits (r2) |
      - Example: COMPR A, S (Compare registers A and S).
   c. **Format 3 (3 bytes)**: | 6 bits (op) | 1 bit (n) | 1 bit (i) | 1 bit (x) | 1 bit (b) | 1 bit (p) | 1 bit (e) | 12 bits (displacement) |
      - `e = 0`.
      - Example: LDA #3 (Load 3 into Accumulator A).
   d. **Format 4 (4 bytes)**: | 6 bits (op) | 1 bit (n) | 1 bit (i) | 1 bit (x) | 1 bit (b) | 1 bit (p) | 1 bit (e) | 20 bits (address) |
      - `e = 1`.
5. **Addressing Modes and Flag Bits**:
   a. **Direct**: `x, b, p = 0`, Operand address is used as is.
   b. **Relative**: Either `b = 1` (Base) or `p = 1` (Program Counter).
   c. **Immediate**: `i = 1, n = 0`, Operand value is in the instruction.
   d. **Indirect**: `i = 0, n = 1`, Operand value points to an address.
   e. **Indexed**: `x = 1`, Adds value of register X to the address.
   - Relative Modes:
     - **Base Relative**: `b = 1, p = 0`, TA = Displacement + (B).
     - **Program Counter Relative**: `b = 0, p = 1`, TA = Displacement + (PC).
6. **Instruction Set**:
   a. Load/Store Register B: LDB, STB.
   b. Floating-Point Arithmetic: ADDF, SUBF, MULF, DIVF.
   c. Register Move: RMO (e.g., RMO S, B).
   d. Register Arithmetic: ADDR, SUBR, MULTR, DIVR.
7. **Input and Output**:
   - Supports all SIC I/O instructions.
   - Additional I/O channels for simultaneous data transfer.
   - Instructions: SIO (Start I/O), TIO (Test I/O), HIO (Halt I/O).

SIC vs. SIC/XE
Basis	**SIC**	**SIC/XE**
Registers	5 registers (A, X, L, SW, PC)	9 registers (A, X, L, SW, PC, B, S, T, F)
Floating-Point Hardware	None	Supported
Instruction Format	Single format	Four formats
Addressing Modes	2 modes	Multiple modes

Assembler Directives
- Definition: Pseudo-instructions that provide instructions to the assembler (not translated into machine code).
- Directives:
  - **START**: Specifies the name and starting address of the program.
  - **END**: Indicates the end of the source program and the first executable statement.
  - **BYTE**: Generates a character or hexadecimal constant.
  - **WORD**: Generates a one-word integer constant.
  - **RESB**: Reserves the indicated number of bytes for a data area.
  - **RESW**: Reserves the indicated number of words for a data area.

Data Movement in SIC and SIC/XE

1. Data Movement in SIC
Instruction	Operand	Description
LDA	EIGHT	Load constant 8 into register A
STA	FIRST	Store in FIRST
LDCH	CHARZ	Load character 'Z' into register A
STCH	C1	Store in character variable C1
- Definitions:
  - FIRST RESW 1: One-word variable.
  - EIGHT WORD 8: One-word constant.
  - CHARZ BYTE C'Z': One-byte constant.
  - C1 RESB 1: One-byte variable.

2. Data Movement in SIC/XE
Instruction	Operand	Description
LDA	#8	Load value 8 into register A
STA	FIRST	Store in FIRST
LDCH	#90	Load ASCII code of 'Z' into register A
STCH	C1	Store in character variable C1
- Definitions:
  - FIRST RESW 1: One-word variable.
  - C1 RESB 1: One-byte variable.
- Notes:
  - Uses immediate addressing with the `#` prefix.

- Character values are represented by ASCII codes (e.g., 90 for 'Z').

Arithmetic Operations in SIC and SIC/XE

1. Arithmetic Operations in SIC
Instruction	Operand	Description
LDA	FIRST	Load FIRST into register A
ADD	INCR	Add value of INCR
SUB	ONE	Subtract 1
STA	SECOND	Store in SECOND
LDA	THIRD	Load THIRD into register A
ADD	INCR	Add value of INCR
SUB	ONE	Subtract 1
STA	FOURTH	Store in FOURTH

- Definitions:
  - FIRST RESW 1: One-word variable.
  - ONE WORD 1: One-word constant.
  - SECOND RESW 1: One-word variable.
  - THIRD RESW 1: One-word variable.
  - FOURTH RESW 1: One-word variable.
  - INCR RESW 1: One-word variable.

2. Arithmetic Operations in SIC/XE
Instruction	Operand	Description
LDS	INCR	Load value of INCR into register S
LDA	FIRST	Load FIRST into register A
ADDR	S, A	Add value of INCR
SUB	#1	Subtract 1
STA	SECOND	Store in SECOND
LDA	THIRD	Load THIRD into register A
ADDR	S, A	Add value of INCR
SUB	#1	Subtract 1
STA	FOURTH	Store in FOURTH

- Definitions:
  - FIRST RESW 1: One-word variable.
  - SECOND RESW 1: One-word variable.
  - THIRD RESW 1: One-word variable.
  - FOURTH RESW 1: One-word variable.
  - INCR RESW 1: One-word variable.

Input/Output Operations in SIC and SIC/XE

1. Input/Output Operations in SIC
Instruction	Operand	Description
TD	INDEV	Test input device

JEQ	INLOOP	Jump to INLOOP if device not ready
RD	INDEV	Read data from input device
STCH	DATA	Store in DATA
- Definitions:
  - INDEV BYTE: Input device code.
  - DATA RESB 1: One-byte variable.

2. Input/Output Operations in SIC/XE
Instruction	Operand	Description
TD	INDEV	Test input device
JEQ	INLOOP	Jump to INLOOP if device not ready
RD	INDEV	Read data from input device
STCH	DATA	Store in DATA
- Additional Features:
  - Supports I/O channels for simultaneous data transfer.
  - Channel Instructions: SIO, TIO, HIO.

Instructions for Creating a PDF
1. **Copy the Text Above**: Copy the cleaned and formatted text into a document editor (e.g., Microsoft Word, Google Docs).
2. **Adjust Formatting**:
   - Use heading styles for section titles (e.g., "Software", "Hardware").
   - Use bullet points and tables as shown.
   - Ensure consistent spacing and alignment.
3. **Export to PDF**:
   - In Microsoft Word: File > Save As > PDF.
   - In Google Docs: File > Download > PDF Document.
4. **Review**: Open the PDF to ensure the formatting is correct.

Let me know if you need further assistance!