@sweta Hardasmalani
C14-2103058

Written Assignment - 1.

. With reference to assembler, consider any assembly program & generate pass-1 & pass-2 output. Also, show content of database table involved in it.

Ans: **Assembly language**: It is a low level programming language that is specific to a particular computer architecture or processor.

Assembler: It is a software tool that translates assembly language code into machine code, which can be executed by a computer's CPU. It performs this in phase 1 & 2.

Two phase assembly process: It is a method used by assembler to translate assembly language source code into machine code. It involves two main processes over the source code:

Pass 1: a) Symbol table generation: The assembler scans the entire source code to identify & record all label, symbol & their corresponding address.

b) Error detection: Pass 1 also involves detecting syntax errors, such as invalid instruction or missing operand.

Pass 2: a) Translation to machine code: Using the information gathered in pass 1, invalid instruction.

b) Output generation: Pass 2 generates the final machine code output along with any necessary metadata.

→ **Assembly program**: program to calculate sum.

```
        ORG 1000    ; Start of program.
START:  LDA NUM1    ; Load accumulator with val
        ADD NUM2    ; Add value of num 2 to accumulator
        STA RESULT  ; Store the result in loc^n RESULT.
        HLT         ; Halt the program.

NUM1    DAT 10      ; First num
NUM2    DAT 20      ; Second num
RESULT  DAT 0       ; Result storage location
```

Pass-1 Output		Pass-2 Output		Src line
Symbol	Address	1000	2001	LDA NUM1
START	1000	1001	3002	ADD NUM2
NUM1	1001	1002	2103	STA RESULT
NUM2	1002	1003	7000	HLT.
RESULT	1003			

- Contents of Database table.

Address	Symbol	Type	Value
1000	START	label	-
1001	NUM1	label	10
1002	NUM2	label	20
1003	RESULT	label	0

- Write a short note on YACC.

→ YACC or Yet Another computer's a powerful tool used in the field of compiler construction to automate the process of generating parsers for programming language. It plays control role in transforming high level language specification, typically written in a formal grammar notation such as BNF into efficient parsers capable of recognizing & analysing the structure of i/p program. YACC functionality & how it fits into the compiler construction process.

1) Formal grammar specification: YACC takes input a formal specification of the syntax rules for a programming language. This specification defines the language syntax in terms of production rules.

which describe how valid sentence or program can be constructed from the language symbol.

2] Lexer Integration: YACC work in conjunction with lexical tool such as lex or flex. Lexer once responsible for breaking down the input text into token or lexemes, which are smallest meaningful unit in the language.

3] Parser Generation: YACC generates a parser for the specific grammar based on the principles of LR parsing. LR parsing is a bottom up parsing technique that built a parse tree for i/p string starting from leaves & working up to the root.

4] Parsing algorithm: YACC parser implements an efficient parsing algorithm that navigate through grammar over input token when there i/p confirms to the syntax.

5] Error handling: YACC parser can be augmented with error handling mechanism to detect & report syntax error in the input. Error recovery strategies can also be determined to help the parse continue parsing after encountering error, ensuring that multiple syntax error in program do not halt the parsing process permanently.

18/3/24 (A+)

Pshita Hara
C14-21035

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

Written assignment-2.

Explain working of direct linking loader with example showing the entries in different database built in DLL

→ A direct link loader is responsible for loading executable program into memory for execution, especially in environment where memory addresses are fixed & absolute. when dealing with dynamic link libraries, the loader needs to handle the loading of both the main program & DLL's it depends on.

Working of direct link loader:

1. Loading the main program: The direct link loader starts by loading the main executable program into memory. This executable contains code & references to external functions & data stored in DLLs.

2. Resolving external references: The loader identifies all the external references made by the main program to function & data in DLLs. These references include function calls & data access. It locates the corresponding addresses of these function & data in the DLLs. These process is known as symbol resolution. If the reference cannot be resolved the loader may generate an error indicating missing symbols or unresolved dependencies

3. Loading DLLs into memory: After resolving external references, the loader loads the required DLLs into memory. It ensures that each DLLs into loaded at any unique address space in memory, avoiding conflicts with other loaded module.

4. Relocating DLL reference: Once the DLLs are loaded, the loader adjust the addresses of the

external references in the main program to point to the correct memory location of the corresponding function & data in the DLL.

5. Setting up execution environment: Before transferring control to main program, the loader setup the execution environment, including stack allocation, initialization of program register & other runtime configuration.

6. Transferring control to the main program: The main program begins execution with access to the function.

- Example: The main program has access to two DLL's.
- Main program ('main.exe'): The main program accesses function & data from two DLLs.

first DLL ('database1.dll')	second DLL ('database2.dll').
Contains funct" & data for first database.	Contains funct" & data for second database.
- Entry 1: Name: "Ishita" Age: 20	- Entry 1 - Product "Laptop" price "₹25,000
- Entry 2: Name: "Ishan" Age: 13	- Entry 2 - Product "ipad" price: "₹1,00,00

When direct link loader loads 'main.exe', it resolves references to function & data in 'database1.dll' & 'database2.dll' - for eg, if the main program calls a function to retrieve data from the first database, the loader ensures that the correct function address is used for the call. Similarly, if the main program accesses data from the second database, the loader resolves the address of that data. The loader transfers control to the entry point of the main program, which then begins execution accessing data from the DLLs as needed.

18/3/24

Rshita Ha
CU-58

Written Assignment-3

Write short note on :

i] Editors & types of editors: Editors are software tools used for creating, modifying & formatting text & code. They play a crucial role in various fields, including software development, content creation & document management. Editors comes in different types, each catering to specific needs & preferences of users.

Types of editors :

1) Text Editor : Text editors are light weight tools primarily used for editing plain text file. Example : Notepad, nano & vim.

2) Code editors : Code editors are specialized tools tailored for editing source code files. They provides advanced feature such as syntax highlighting, well folding & debugging support. Code editor are normally used in software development for writing & managing code efficiently.

3) Integrated development environment (IDEs) : They are comprehensive software package that combines code editors with other development tools & features. They offer feature like project management, debugging, version control & built in compiler or interpreter. Example : VS code, Pycharm, Eclipse, etc.

4) WYSIWYG Editor : They enable user to create & edit document with a visual representation of the final output. They allow user to format text, insert image & design layout

without needing to understand underlying markup or code. Example: wordpress, microsoft word.

5) version control editors: They are specially integrated with version control system like git. They provide feature for managing version, history, committing changes & resolving merge conflicts. Example: Github Desktop.

6) Terminal Editor: Terminal editor are text editors designed to work within a CLI. They are light weight & efficient, making them suitable for server based editing.
Eg: Vim, Emacs, Nano.

Backpatching in intermediate code generation.

→ It is a technique used in intermediate code generation, particularly in the context of compilers. It facilitate the handling of jumps or branch in code generation when the destination address is not known during initial generation of code. Backpacking in intermediate code generation:
In intermediate code generation, compiler often encounter situation where the target address of a jump or branch statement is not immediately known. This typically happens due to dealing with additional statement or loop constructs.
Steps in backpatching:

1. Placeholder creation: During intermediate code generation, when a jump or branch statement is encountered with an unknown target address,

a placeholder is created to represent target location.

2. Recording placeholder information: The compiler maintain a DS to record information about each placeholder.
3. Determination of target address: As the compilation process progress, the compiler determined the actual address of the target location association with placeholder.
4. Updating placeholder addresses: Once the target address are known, the compiler performs backpacking by updating placeholder in the intermediate code with actual address.
5. Completion of intermediate code generation: After backpacking is complete, the intermediate code is finalized & all jump once branch statement now reference valid target address.

- **Benefits:**
1. Flexibility: Backpatching allows for the generation of intermediate code without the need to know target address upfront.
2. Efficiency: It simplifies the handling of jumps & branches in the intermediate code generation.
3. Modularity: It facilitates modern approach to intermediate code generation.

8/3/24