Computer Organization

605.204

Module Four

Part Four

Additional Features

Module Four

- Part Four
- In this presentation, we are going to talk about :
- Assembler
  - Additional Features

Previously

- Previously we talked about:
- Assembler Object file
- Now: Additional Assembler Features

Other Assembler Features

- Assemblers have many of these characteristics:
  - Relocation
  - Literal values
  - Pseudo Instructions
  - Symbol defining directives
  - Program Blocks
  - Control Sections

Program Relocation

- Small special processors
  - Space for no more than one program
- Large memory systems
  - Space for several programs
- Need to provide information to loader
- Address issue

The Assembler Knows !

- Absolute value - data constants
- Relative value - instruction address, data address
- Relocatable Program - has information needed to change the addresses
- Relocation Section of Object Program file
  - Tells Loader which fields need to be modified
  - and by what amount

Literal Values

- Programmers want to specify literal constant values as they write the program. Allow the assembler to determine storage location.
- Not the same as Immediate operand.
- Character and hex constant values
  - .byte directive alternative
- Literal Pool - memory area where literals are stored
- **LTORG** - assembler directive, denotes location create a literal memory pool at this address

Literal Values

- Character values =C'END'
- Hex values =X'0A'
- LITTAB - Assembler table

  name, operand value, length, assigned address
- A question of 'duplicate' literals
  - =C'END' =X'454E44' Same ?

Pseudoinstructions

Hardware Software Interface

Some assemblers also implement *pseudoinstructions*, which are instructions provided by an assembler but not implemented in hardware. Chapter 3 contains many examples of how the MIPS assembler synthesizes pseudoinstructions and addressing modes from the spartan MIPS hardware instruction set. For example, section 3.5 in Chapter 3 describes how the assembler synthesizes the `blt` instruction from two other instructions: `slt` and `bne`. By extending the instruction set, the MIPS assembler makes assembly language programming easier without complicating the hardware. Many pseudoinstructions could also be simulated with macros, but the MIPS assembler can generate better code for these instructions because it can use a dedicated register (`$at`) and is able to optimize the generated code.

Pseudoinstructions (continued)

- Feature for the programmer
- Assembler makes true instructions
  - Small number
  - No branch
- No side effects
  - Can only modify $at register
  - No changes to stack or other registers

Symbol Defining Directives

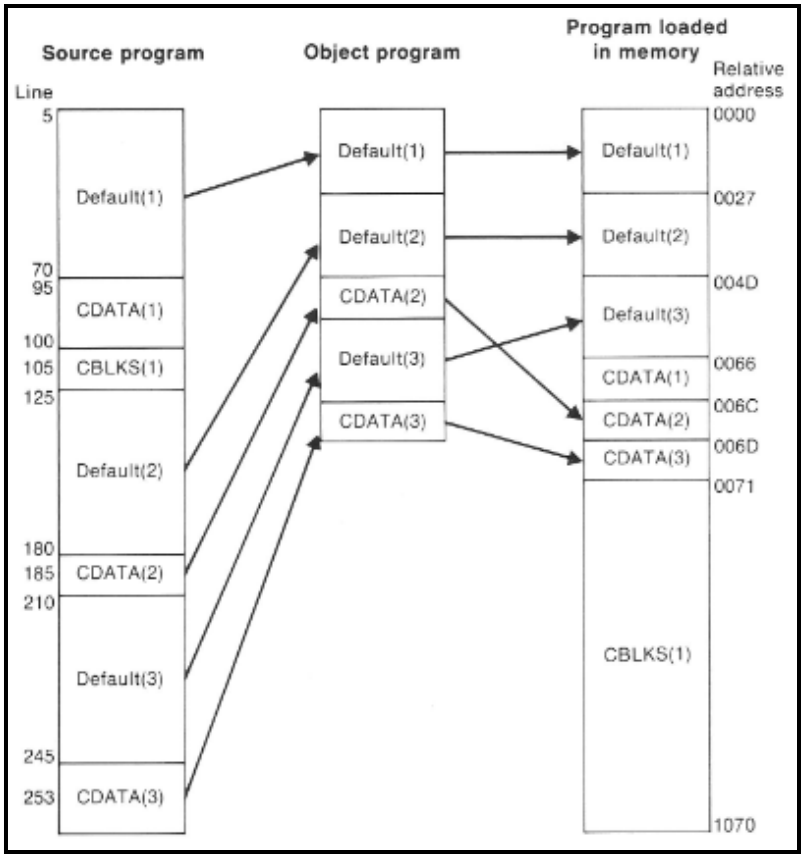
- Programmers need to have greater control of assembly process.
- **EQU** - Directly assign values to program symbols
- **ORG** - Set the LOCCTR value to value of symbol
- Operand Expressions
  - EQU *****
  - EQU BUFEND **-** BUFFER

  Relative value

  Absolute value

Program Blocks

- **USE** directive
- Multiple LOCCTR values
- Change the order of the object code
- Result is blocks of code and data that are loaded separately
- Code block, data block, I / O buffers



Control Sections

- Independent program
  - Subroutine, library routine
- **CSECT** Directive
- Separately assembled
- New Directives
  - **EXTREF** reference to outside section
  - **EXTDEF** defined in this section
- Uses the Symbol records of the Object program
- Linked by the Loader to other sections to make program.

Design Options

- One-Pass
  - Problem: Resolve the forward references
- Define all data objects before they are referenced
- Object program made directly into memory - Load and Go
- Object program file
  - Use additional Text records
- Multi-Pass
  - As many as it takes

One Pass

- Two passes to over come the ‘forward reference issue’
- The In-core assembler
  - Keeps a copy of the object program in memory
  - Then executes – Assemble and Go
  - Student programming exercises
- One pass is fast, really a one and one half pass
  - Fix up forward references when symbols are discovered.
- Can also be restrictive
  - All data labels defined first
  - Only subset of addressing or pseudo instructions allowed

Summary

- Additional Assembler features
  - Relocation
  - Literal values
  - Symbol defining directives
  - Pseudo Instructions
  - Program Blocks
  - Control Sections

- This week we talked about - the Assembler
  - Function, Organization, and Features