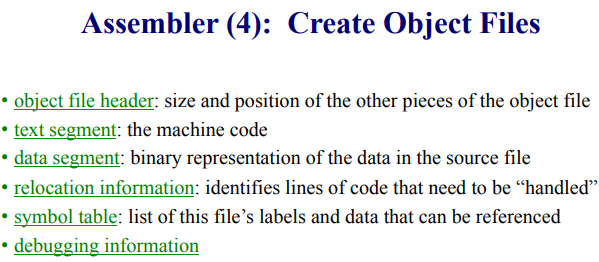
**TAL->MAL:**

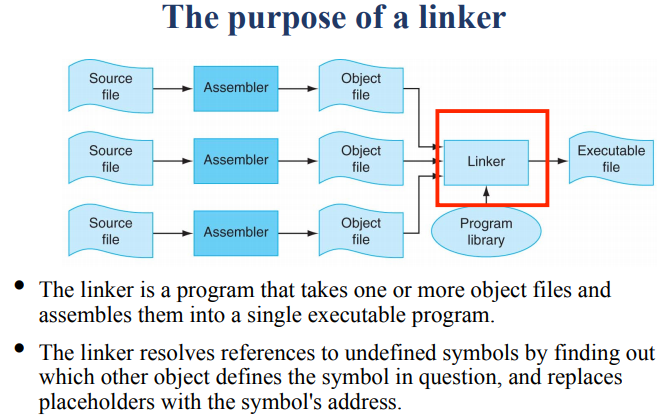
**R format**- Register -> Register Comparison:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 6 Bit Opcode | 5 Bit Source Register #1 | 5 Bit Source Register #2 | 5 Bit Destn. register | 6 Bit Shamt | 6 Bit Func |

**I format**-> LW/SW/Branch/Immediate values:

* Branch Case

|  |  |  |  |
| --- | --- | --- | --- |
| 6 Bit Opcode | 5 Bit Source Register #1 | 5 Bit Source Register #2 | 16 Bit Offset |

* Immediate Case

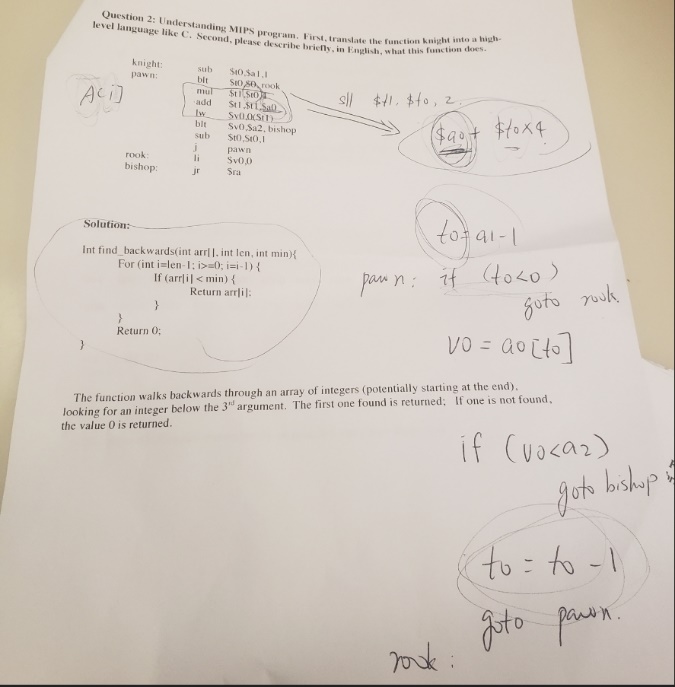
|  |  |  |  |
| --- | --- | --- | --- |
| 6 Bit Opcode | 5 Bit Source Register #1 | 5 Bit Destn. Register | 16 Bit Immediate |

**J format**- Jump:

|  |  |
| --- | --- |
| 6 Bit Opcode | 26 Bit EXACT address |

* NOTE: First 4 and Last 2 bits are set to zero

**Concepts:**

* **Steps to starting a program:**
  + Write Code->Compiler->Assembler->Object File Code (Binary)->Linker->Exe->Loader->Memory
    - Compiler translates high level code to Assembly
    - Assembler reads and uses directives, replaces pseudo-code, creates MAL, and creates Object File
    - Linker combines several object (.o) files into a single executable, it stiches files together and resolves references
    - Loader’s job is to load exe files into memory
* **Chronological Order for making->Executing C Code**
  + Student writes code in C
  + C code is translated into MIPS
  + MAL translated into TAL
  + Link tables produced
  + Code and data stitched together
  + Links are edited
  + Space in memory is reserved
  + Execution begins at main

**MIPS->C:**

* Translate line by line with pseudo code
* Translate that to C code
* In this example, lines 3,4,5=A[0+i]
  + T0=i
  + T1=A[0]

