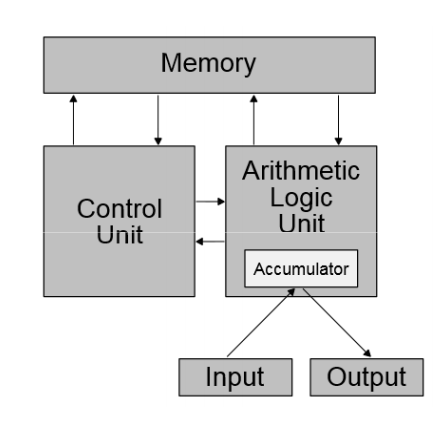
Turing Machine - capaz de correr qualquer algoritmo matemático

Turing Complete - capacidade equivalene a uma máquina de Turing

ENIAC - Primeiro computador Turing Complete

Stored Program Computer - guarda as instruções do programa na memória (Alan Turing)

Arquitetura von Neumann – tal como Stored Program Computer



Transístor – on/off switch controlado por eletricidade (quando comparado com tubos de vácuo são mais pequenos e precisam de menos energia, então aquecem mesmo)

Computadores com transístores – podem conter milhares de circuitos binários num espaço compacto

Circuito integrado – combina um conjunto de transístores numa peça pequena

Microprocessadores – incorpora as funções da unidade central de processamento num único circuito integrado

Lei de Moore – o número de transístores num circuito integrado denso duplica a cada 18 meses (é uma observação e projeção tendo em conta a tendência da evolução)

**Tradução e início do programa**

- Compilador

- Assembler

- Linker

- Loader

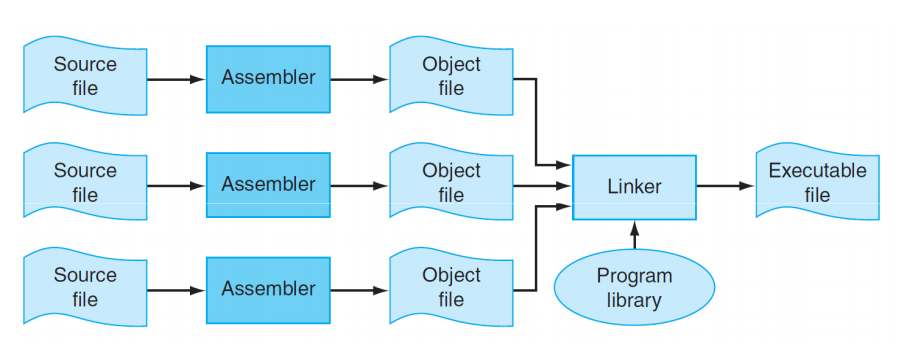
Alguns sistemas combinam estes passos para reduzirem o tempo de tradução, mas estes são os 4 passos lógicos por onde o programa passa

Compiler – Transforma uma linguagem de alto nível numa linguagem de programa assembly, uma forma simbólica para que a máquina entenda

Assembler – Transforma o programa da linguagem assembly num ficheiro objeto que é a combinação entre a linguagem da máquina e as instruções e dados da máquina e a informação necessária para colocar devidamente as instruções na memória

Linker – Combina independentemente os ficheiros objetos e resolve todas as labels indefinidas num executável

Loader – coloca o ficheiro executável na memória main para que esteja pronto a executar



Linguagens de alto nível – código é curto e limpo

Programa assembly – código mais difícil de seguir, porque muitas operações simples são necessárias para cumprir tarefas simples, pode ser ainda mais difícil de seguir por causa de referências a localizações na memória

Código máquina – com muito esforço, poderia-se usar o opcode e o formato de cada instrução para traduzir o código para a anterior

Address Binding – processo que mapeia um espaço de endereçamento noutro

- Address do Source code normalmente é simbólico (var xpto)

- No compiler/assembler conecta os address simbólicos para address recolocáveis (604 bytes a partir do início deste módulo)

- Linker/Loader conecta address recolocáveis para address recolocáveis (address 0x0FFF0904)

**Ficheiro Objeto**

Normalmente contém 6 partes diferentes

- Cabeçalho – descreve o tamanho e a posição das outras 5 partes

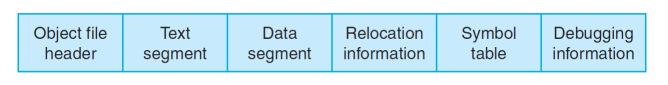
- Segmento de texto – contém o código máquina

- Segmento Static data – contém a data alocada para o programa

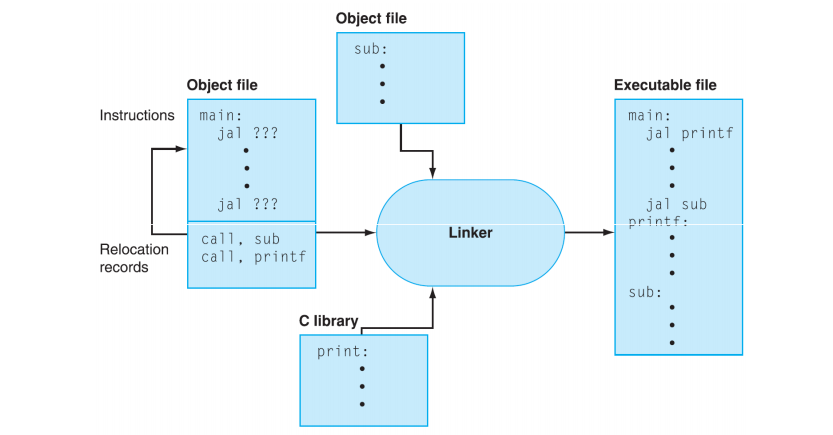
- Informação de recolocação – identifica instruções e palavras data que dependem de address absolutos quando o programa é carregado para a memória

- Tabela de símbolos – contém as labels restantes que não estão definidas, tais como definições globais e referências externas

- Informação de Debug – contém uma descrição de como os módulos foram compilados para que um debugger possa associar as instruções da máquina com ficheiros C source e fazer estruturas de dados legíveis



**Linking**



O linker produz um ficheiro executável que tem o mesmo formato que um ficheiro objeto, exceto que não contém referências por resolver ou recolocar informação

O linker normalmente inclui estes 3 passos

- Encontrar bibliotecas usadas pelo programa

- Juntar segmentos ao colocar código e módulos de dados simbolicamente na memória e a recolocar as suas instruções ao ajustar as referências absolutas

- Resolver referências entre ficheiros

**Loader**

O loader normalmente inclui estes 6 passos para carregar um ficheiro executável na memória

- Ler o cabeçalho do ficheiro para saber os segmentos de dados

- Criar espaço de endereços grandes o suficiente para o texto e os segmentos de dados

- Copiar as instruções e os dados do ficheiro executável para a memória

- Copiar os argumentos do programa para a pilha

- Inicializar os registos da máquina e colocar o apontador da pilha no top da pilha

- Saltar para a rotina de arranque que copia os argumentos do programa para os argumentos do registo e chama a rotina principal do programa.

**Dynamic Linking**

Tem desvantagens mesmo sendo a forma mais rápida de chamar bibliotecas, essas bibliotecas tornam-se parte do código executável e carrega todas as rotinas da biblioteca que são chamadas em qualquer sítio do executável. Estas desvantagens levam às DLL

Permite uma melhor utilização memory-space

**Binary**

**Normal binary with and without sign**

**2 complement**

**Binary Addition (overflow)**

**Floating Point numbers (Single and Double Precision, 32 and 64 bits)**

**IEEE754 (bias 127 single precision, 1023 double precision)**

**Floating Point addition**

**Rounding with IEE754**

**MIPS Registers**

**Algoritmo de adição em vírgula flutuante para números em formato IEEE754:**

**1. Igualar os expoentes deslocando a vírgula do menor valor menor**

**2. Fazer a adição**

**3. Normalizar o resultado**

**4. Verificar se temos overflow/underflow (expoente não representável)**

**5. Arredondar ao número de bits**

**6. Caso não esteja normalizado voltar ao passo 3**