Simulador Kands - Linguagem de Máquina e Linguagem Assembly (Montagem)

É importante notar que a linguagem de máquina de hardware e as configurações associadas com nosso simulador são muito mais simples do que os de qualquer computador real. Uma linguagem de máquina real pode abranger dezenas ou centenas de instruções, e uma CPU real pode conter centenas ou milhares de componentes configuráveis. No entanto, o nosso modelo é suficiente para demonstrar o comportamento de um computador em seu nível mais baixo.
O simulador é também útil na representação da dificuldade e tédio de programação numa linguagem de máquina. Nos últimos 50 anos, a programação de computadores tem avançado de forma significativa, e a maioria dos programadores modernos são capazes de evitar a programação direta em linguagem de máquina. Algumas das primeiras ferramentas de programação eram linguagens de montagem, que substituem palavras para padrões de bits, permitindo ao programador escrever:

ADD R0 R1 R2

em vez da instrução em linguagem de máquina:

1010000100000110

É muito mais fácil para os programadores lembrar e compreender instruções em linguagem assembly do que os padrões de 0s e 1s. Além disso, a maioria das linguagens de montagem suportam o uso de nomes de variáveis usadas pelo sistema operacional, permitindo que os programadores especifiquem posições de memória por nomes descritivos, ao invés de endereço numérico. Isso simplifica muito a tarefa do programador, como ela não precisa mais se preocupar com a localização física dos dados, ou seja, com os endereços dos locais da memória.
No simulador Kands, o usuário pode entrar instruções em linguagem assembly diretamente na memória. O modo padrão para exibir o conteúdo de locais de memória, identificado como "Auto" na caixa *View As*, irá reconhecer automaticamente instruções na linguagem de montagem e vai exibi-las como texto (mudando o rótulo de "Inst" para reconhecer que são as instruções).
Depois de introduzir as instruções, o usuário pode alternar entre a visualização de instruções de montagem ou de linguagem de máquina no formulário selecionando Inst (para instruções em linguagem assembly) ou 2 (para instruções em linguagem de máquina em formato binário) na caixa *View* à esquerda da instrução.

Instrução em linguagem de máquina	**Exemplo**	**Significado**	**Linguagem de montagem (Assembly)**
1 000 0001 0 RR MMMMM	1 000 0001 0 10 01101	R2 = MM[13]	LOAD R2 13
1 000 0010 0 RR MMMMM	1 000 0010 0 11 01000	MM[8] = R3	STORE 8 R3
1 001 0001 0000 RR RR	1 001 0001 0000 10 00	R2 = R0	MOVE R2 R0
1 010 0001 00 RR RR RR	1 010 0001 00 11 10 01	R3 = R2 + R1	ADD R3 R2 R1
1 010 0010 00 RR RR RR	1 010 0010 00 11 01 00	R3 = R1 - R0	SUB R3 R1 R0
1 010 0011 00 RR RR RR	1 010 0011 00 00 11 01	R0 = R3 & R1	AND R0 R3 R1
1 010 0100 00 RR RR RR	1 010 0100 00 10 10 11	R2 = R2 | R3	OR R2 R2 R3
0 000 0001 000 MMMMM	0 000 0001 000 01010	PC = 10	BRANCH 10
0 000 0010 000 MMMMM	0 000 0010 000 00010	if Zero Flag set, PC=2	BZERO 2
0 000 0011 000 MMMMM	0 000 0011 000 00111	if Neg. Flag set, PC=7	BNEG 7
0000 0000 0000 0000		no operation	NOP
1111 1111 1111 1111		halt execution	HALT

Figura 14.14 Instrução da linguagem de montagem (Assembly).

Exemplos:

```
int main() {
    int a = 9;
    int b = 1;
    int c = 0;

    c = a + b;
    }
}
```

```
LOAD R0 5               // carrega posicao de memoria 5 em R0
LOAD R1 6               // carrega posicao de memoria 6 em R1
ADD R2 R0 R1            // adiciona R0 e R1, armazena em R2
STORE 7 R2              // armazena R2 na posicao de memoria 7
HALT                    // parada
9                       // dado a ser somado: 9
1                       // dado a ser somado: 1
0                       // posicao onde a soma eh armazenada

1000000100000101        // carrega posicao de memoria 5 em R0
1000000100100110        // carrega posicao de memoria 6 em R1
1010000100100001        // adiciona R0 e R1, armazena em R2
1000001001000111        // armazena R2 na posicao de memoria 7
1111111111111111        // parada
0000000000001001        // dado a ser somado: 9
0000000000000001        // dado a ser somado: 1
0000000000000000        // posicao onde a soma eh armazenada
```

```
int main(){
    int a = 1;
    int b = 2;
    int c = 0;

    c = a + b;

    if (c > 1){
        c = 10;
    }
    else{
        c = 100;
    }
}
```

```
LOAD R0 14
LOAD R1 15
ADD R2 R0 R1
STORE 16 R2
LOAD R3 17
SUB R3 R2 R3
BNEG 11
BZERO 11
```

LOAD R3 18
STORE 16 R3
BRANCH 13
LOAD R3 19
STORE 16 R3
HALT
1
2
0
1
10
100
0

1000000100001110
1000000100101111
1010000100100001
1000001001010000
1000000101110001
1010001000111011
0000001100001011
0000001000001011
1000000101110010
1000001001110000
0000000100001101
1000000101110011
1000001001110000
1111111111111111
0000000000000001
0000000000000010
0000000000000000
0000000000000001
0000000000001010
0000000001100100
0000000000000000