Tradutor de Linguagem de Máquina do processador MIPS

- Instruções suportadas
- Tipo de Instruções MIPS
- Implementação
- Limitações

Instruções Suportadas

- sll
- srl
- jr
- add
- sub
- and
- or
- slt
- beq
- bne
- addi
- lw
- sw
- j
- jal

Tipo de Instruções MIPS – Tipo R

Codificação:

000000SSSSSTTTTTDDDDDQQQQQFFFFFF

Sigla	Legenda
S	Registrador S
T	Registrador T
D	Registrador D
Q	Quantidade de bits das operações de deslocamento
F	Código da Função

Instrução	F	F (Binário)	Exemplo
sll	0	000000	sll S, D, Q
srl	2	000010	srl S, D, Q
jr	8	001000	jr S
add	32	100000	add D, S, T
sub	34	100010	sub D, S, T
and	36	100100	and D, S, T
or	37	100101	or D, S, T
slt	42	101010	slt D, S, T

Tipo de Instruções MIPS – Tipo I

Codificação:

OOOOOOSSSSSDDDDDIIIIIIIIIIIIIIII

Sigla	Legenda
O	Código de Operação
S	Registrador S
T	Registrador D
I	Imediato de 0 até 65535

Instrução	O	O (Binário)	Exemplo
beq	4	000100	beq D, S, I
bne	5	000101	bne D, S, I
addi	8	001000	addi D, S, I
lw	35	100011	lw D, I(S)
sw	43	101011	sw D, I(S)

Tipo de Instruções MIPS – Tipo J

Codificação:

OOOOOOEEEEEEEEEEEEEEEEEEEEEEEEEE

Sigla	Legenda
O	Código de Operação
E	Endereço: Imediato de 0 até 67108863

Instrução	O	O (Binário)	Exemplo
j	2	000010	j E
jal	3	000011	jal E

Implementação: Organização do código

O programa foi dividido em três bibliotecas para uma melhor organização:

- libMips.h contém as funções responsáveis pela tradução em si.
- MIPSconst.h contém as constantes necessárias para a tradução.
- arrayUtils.h contém funções para auxiliar operações com vetores.

O programa principal tradutorMIPS.c é responsável apenas pela entrada e saída.

Implementação:
Fluxo da tradução

O programa principal, faz a leitura do arquivo de entrada, linha por linha.

A biblioteca libMips.h divide essa *string* em quatro (instrução, operando1, operando2 e operando3).

Através do nome da instrução é identificado o seu tipo, e consequentemente o seu padrão de codificação.

Os bits são então dispostos seguindo esse padrão.

O programa principal salva esses bits no arquivo de saída.

Implementação:
Detalhes da Implementação

A biblioteca MIPSconst.h possui vetores constantes com os nomes e valores dos registradores, instruções e código de operação.

Os vetores com os nomes são utilizados para a identificação.

Uma vez que o registrador (ou instrução) é encontrado num vetor, os bits referentes ao valor do mesmo se encontram no mesmo índice de um vetor auxiliar.

Implementação:
Detalhes da Implementação

Exemplo:

```
//Instruções Tipo J

const char J_MNE[2][4] = {"j", "jal"};

const unsigned char J_OPCODE[5][6] =

  {0, 0, 0, 0, 1, 0, //02 = j

   0, 0, 0, 0, 1, 1};//03 = jal
```

Implementação:
Detalhes da Implementação

As instruções foram separadas em três duplas de vetores.
Uma dupla contém os dados referentes as instruções suportadas do Tipo R, outra do Tipo I e a última do tipo J.
Existe também uma dupla referente ao registradores.

Implementação:
Detalhes da Implementação

- Se a *string* contendo a instrução não é encontrada em nenhum dos três vetores um erro de instrução não suportada é gerado.
- Se é esperado que um operando seja um registrador e ele não é encontrado no vetor de registradores, um erro de registrador inválido é gerado.
- Se um imediato superar o tamanho máximo que quantidade de bits onde ele será alocado suporta um erro de imediato muito grande é gerado.

Implementação:
Detalhes da Implementação

Com a instrução identificada e os operandos validados a função de codificação é chamada.

*void instrucaoTipoR(const unsigned char *rs, const unsigned char *rt, const unsigned char *rd, unsigned int shamt, const unsigned char *func)*

*void instrucaoTipoI(const unsigned char *opCode, unsigned char *rs, unsigned char *rt, unsigned int imediato)*

*void instrucaoTipoJ(const unsigned char *opCode, unsigned int imediato)*

Implementação:
Detalhes da Implementação

As funções de codificação alocam os bits passados como parâmetro na devida posição, segundo o padrão de codificação de cada tipo de instrução.

O conjunto de bits já codificados é armazenado num vetor.

Implementação:
Detalhes da Implementação

Para representar os conjuntos de bits foi escolhido a utilização de vetores de *unsigned char*.

Esse tipo de dados foi escolhido ao invés do *char*, pois o *unsigned char* não possui sinal, e só iremos armazenar os valores 0 ou 1.
Não optamos por utilizar strings '0' (48) ou '1' (49) no armazenamento, apenas na saída, pois o valor na forma de caracter ASCII não representa o dado de verdade.

Limitações

- Conjunto reduzido de instruções
- Falta de um analisador sintático