Especificações Técnicas do microRISC-8

Versão 1.0

Arquitetura Geral

- **Tipo:** RISC (Reduced Instruction Set Computer).
- **Bits:** 8 bits.
- **Clock Estimado:** 2 MHz a 4 MHz.
- ***Clock Real:*** *Igual ao clock do processador que o emula.*
- **Registradores:**
  - 14 registradores de uso geral (R1-R14): 8 bits.
  - Acumulator (ACC ou R0): 8 bits.
  - Registrador Zero (RZ ou R15): 8 bits. (Valor fixo de: 0x0000).
  - Program Counter (PC): 16 bits. (Por padrão vale: 0x0000).
  - Instruction Register (IR): 8 bits.
  - Status Register (SR): 8 bits (Flags detalhadas abaixo).
  - Stack Pointer (SP): 16 bits. (Pilha decrescente. Valor padrão: E000).
  - Global Pointer (GP): 16 bits. (Por padrão vale: 0xC000).
  - Memory Address Register (MAR): 16 bits.
  - Memory Data Register (MDR): 8 bits.

Registrador de Status (SR)

Bit	Nome	Descrição
1	Z	Zero
0	C	Carry

- Zero (Z): Setado quando o resultado da operação for zero.
- Carry (C): Setado quando há overflow/carry em operações aritméticas.

Mapeamento dos Registradores (R0 [ACC], R1 a R14, e RZ[R15])

Os registradores são mapeados da seguinte maneira:

Registrador	Binário (4 bits)
R0 (ACC)	0000
R1	0001
R2	0010
R3	0011
R4	0100
R5	0101
R6	0110
R7	0111
R8	1000
R9	1001
R10	1010
R11	1011
R12	1100
R13	1101
R14	1110
R15 (RZ)	1111

Mapeamento de Memória (64 KB)

Endereço	Dispositivo	Descrição
0x0000 - 0xBFFF	ROM	48 KB (49.152 bytes)
0xC000 - 0xDFFF	RAM	8 KB (8.192 bytes)
0xE000 - 0xFFBF	*Para Expansão*	*~ 8 KB (8.125 bytes)*
0xFFC0 - 0xFFFF	I/O	64 B (64 bytes)

- O ponteiro da pilha sempre começa em 0xE000, ou seja, um byte a mais do fim da RAM, mas não é um problema, pois a CPU sempre faz um decremento antes de inserir algo na pilha, ou seja, a base da pilha ficará em 0xDFFF.
- O endereço FFFE e FFFD estão reservados para INPUT de inteiros e STATUS do input de inteiros, respectivamente.
- O endereço FFFC e FFFB estão reservados para INPUT de caracteres e STATUS do input de caracteres, respectivamente.
- O endereço FFFF está reservado para OUTPUT.

Periféricos Integrados

- **I/O (Entrada/Saída):**
  - Suporte a 1 teclado e 1 monitor de saída.

Operandos e Modos de Endereçamento

- Registrador. Ex: R1
- Imediato. Ex: #5 *ou* #-7
- Char. Ex: ‘c’
- Endereçamento direto. Ex: &20
- Label. Ex: CALL SOMA

Organização dos Opcodes

Cada opcode possui 8 bits, permitindo 256 opcodes possíveis. A estrutura do opcode é dividida da seguinte forma:

- **4 bits para mapeamento dos registradores** (o que inclui o acumulador e os 15 registradores de uso geral).
- **8 bits para valores imediatos e chars**, que exigem um ciclo de clock adicional.
- **16 bits para endereços de memória**, com 2 ciclos de clock implícitos.

Palavra de bits:

- [opcode (8 bits)]
- [opcode (8 bits)] [operando (08 bits)]
- [opcode (8 bits)] [endereço (16 bits)]
- [opcode (8 bits)] [registradores (4 + 4 bits)]
- [opcode (8 bits)] [registrador] [operando (08 bits)]
- [opcode (8 bits)] [registrador] [endereço (16 bits)]

Mapeamento dos Opcodes de 0 a 41

Opcode	Instrução	Ciclos (+ implícitos)
0	**LDA** (LOAD ADDRESS)	4
1	**LDI** (LOAD IMMEDIATE)	3
2	**STA** (STORE ADDRESS)	4
3	**MOV** (MOVE)	2
4	**ADD** (ADDITION)	2
5	**ADD.I** (ADDITION IMMEDIATE)	3
6	**SUB** (SUBTRACTION)	2
7	**SUB.I** (SUBTRACTION IMMEDIATE)	3
8	**SHT.L** (SHIFT LEFT LOGICAL)	2
9	**SHT.R** (SHIFT RIGHT LOGICAL)	2
10	**HLT** (HALT PROGRAM)	1
11	**JMP** (JUMP)	3
12	**CMP** (COMPARE)	2
13	**CMP.I** (COMPARE IMMEDIATE)	3
14	**BEQ** (BRANCH IF EQUAL)	3
15	**BNE** (BRANCH IF NOT EQUAL)	3
16	**BLE** (BRANCH IF LESS OR EQUAL)	3
17	**BGE** (BRANCH IF GREATER OR EQUAL)	3
18	**BLT** (BRANCH IF LESS THAN)	3
19	**BGT** (BRANCH IF GREATER THAN)	3
20	**BEQ.R** (BRANCH IF EQUAL RELATIVE)	2
21	**BNE.R** (BRANCH IF NOT EQUAL RELATIVE)	2
22	**BLE.R** (BRANCH IF LESS OR EQUAL RELATIVE)	2
23	**BGE.R** (BRANCH IF GREATER OR EQUAL RELATIVE)	2
24	**BLT.R** (BRANCH IF LESS THAN RELATIVE)	2
25	**BGT.R** (BRANCH IF GREATER THAN RELATIVE)	2
26	**AND** (AND OPERATION BETWEEN TWO REGS)	2
27	**OR** (OR OPERATION BETWEEN TWO REGS)	2
28	**XOR** (XOR OPERATION BETWEEN TWO REGS)	2
29	**NOT** (NOT OPERATION IN REG)	2
30	**END** (END PROGRAM)	1
31	**IN** (GET INPUT DATA)	1
32	**DRAW** (SET DRAW CHAR)	1
33	**OUT** (GET OUTPUT INTEGER)	1
34	**CALL** (CALL SUBROTINE)	3
35	**RET** (RETURN SUBROTINE)	1
36	**INI.P** (INITIALIZE POINTER)	3
37	**SET.P** (SET POINTER)	2
38	**GET.P** (GET POINTER)	2
39	**UPD.P** (UPDATE POINTER)	2
40	**UPI.P** (UPDATE IMMEDIATE POINTER)	2
41	**GETC** (GET CHAR INPUT)	1

Exemplos de Instruções em Assembly e Representação Binária

Aqui estão exemplos de como as instruções podem ser representadas em Assembly e em binário:

LDA (LOAD ADDRESS)

- **Exemplo em Assembly:** LDA R1, 0x20
- **Significado:** Carrega o valor do endereço de memória 0x20 no registrador R1.
- **Formato do Opcode:** LDA <registrador>, <endereço de memória>
- **Representação Binária:** 0000 0000 0000 0001 0000 0000 0010 0000
    - 0000 0000: Opcode LDA
    - 0000 0001: Registrador R1
    - 0000 0000 0010 0000: Endereço 0x20

LDI (LOAD IMMEDIATE)

- **Exemplo em Assembly:** LDI R1, #100
- **Significado:** Carrega o valor imediato 100 no registrador R1.
- **Formato do Opcode:** LDI <registrador>, <valor imediato>
- **Representação Binária:** 0000 0001 0000 0001 0110 0100
    - 0000 0001: Opcode LDI
    - 0000 0001: Registrador R1
    - 0110 0100: Valor imediato 100

STA (STORE ADDRESS)

- **Exemplo em Assembly:** STA R1, 0x30
- **Significado:** Armazena o valor do registrador R1 no endereço de memória 0x30.
- **Representação Binária:** 0000 0010 0000 0001 0000 0000 0011 0000
    - 0000 0010: Opcode STA
    - 0000 0001: Registrador R1
    - 0000 0000 0011 0000: Endereço 0x30

MOV (MOVE)

- **Exemplo em Assembly:** MOV R1, R2
- **Significado:** Move o valor do registrador R2 para o registrador R1.
- **Representação Binária:** 0000 0011 0001 0010
  - 0000 0011: Opcode MOV
  - 0001: Registrador R1
  - 0010: Registrador R2

ADD (ADDITION)

- **Exemplo em Assembly:** ADD R1, R2
- **Significado:** Adiciona o valor do registrador R2 ao registrador R1.
- **Representação Binária:** 0000 0100 0001 0010
  - 0000 0100: Opcode ADD
  - 0001: Registrador R1
  - 0010: Registrador R2

ADD.I (ADDITION IMMEDIATE)

- **Exemplo em Assembly:** ADD.I R1, #1
- **Significado:** Adiciona o valor do imediato 1 ao registrador R1.

SUB (SUBTRACTION)

- **Exemplo em Assembly:** SUB R1, R2
- **Significado:** Adiciona o valor do R2 ao R1.

SUB.I (SUBTRACTION IMMEDIATE)

- **Exemplo em Assembly:** SUB.I R1, #2
- **Significado:** Adiciona o valor do imediato 2 ao R1.

SHT.L (SHIFT LEFT LOGICAL)

- **Exemplo em Assembly:** SHT.L R1, R2
- **Significado:** Faz um deslocamento lógico à esquerda no R1, sendo realizado X vezes, definido pelo valor dentro de R2. Exemplo: Se o R1 for 2 e R2 for 1, fazer o SHT.L equivale a multiplicar R1 por 2, ou seja, R1 será igual a 4. Isso equivale a multiplicar por potencias de 2.

SHT.R (SHIFT RIGHT LOGICAL)

- **Exemplo em Assembly:** SHT.R R2, R3
- **Significado:** Faz um deslocamento lógico à direita no R2, sendo realizado X vezes, definido pelo valor dentro de R3. Exemplo: Se o R2 for 4 e R3 for 1, fazer o SHT.R equivale a dividir R2 por 2, ou seja, R2 será igual a 2. Isso equivale a dividir por potencias de 2.

HLT (HALT PROGRAM)

- **Exemplo em Assembly:** HTL
- **Significado:** Pausa a execução até que se tenha uma interrupção, como pressionar uma tecla no teclado ou algo parecido.

JMP (JUMP)

- **Exemplo em Assembly:** JMP 0x20
- **Significado:** Salto incondicional para o endereço especificado (0x20).

CMP (COMPARE)

- **Exemplo em Assembly:** CMP R3, R4
- **Significado:** Compara dois registradores e altera o SR (Zero e Carry) (se necessário).

CMP.I (COMPARE IMMEDIATE)

- **Exemplo em Assembly:** CMP.I R3, 5
- **Significado:** Compara a registrador com um valor imediato e altera o SR (Zero e Carry) (se necessário).

BEQ (BRANCH IF EQUAL)

- **Exemplo em Assembly:** BEQ 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BNE (BRANCH IF NOT EQUAL)

- **Exemplo em Assembly:** BNE 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BLE (BRANCH IF LESS OR EQUAL)

- **Exemplo em Assembly:** BLE 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BGE (BRANCH IF GREATER OR EQUAL)

- **Exemplo em Assembly:** BGE 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BLT (BRANCH IF LESS THAN)

- **Exemplo em Assembly:** BLT 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BGT (BRANCH IF GREATER THAN)

- **Exemplo em Assembly:** BGT 0x20
- **Significado:** Salto condicional para o endereço especificado (0x20).

BEQ.R (BRANCH IF EQUAL RELATIVE)

- **Exemplo em Assembly:** BEQ.R -5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (-5).

BNE.R (BRANCH IF NOT EQUAL RELATIVE)

- **Exemplo em Assembly:** BNE.R 5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (5).

BLE.R (BRANCH IF LESS OR EQUAL RELATIVE)

- **Exemplo em Assembly:** BLE.R -5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (-5).

BGE.R (BRANCH IF GREATER OR EQUAL RELATIVE)

- **Exemplo em Assembly:** BGE.R -5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (-5).

BLT.R (BRANCH IF LESS THAN RELATIVE)

- **Exemplo em Assembly:** BLT.R -5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (-5).

BGT.R (BRANCH IF GREATER THAN RELATIVE)

- **Exemplo em Assembly:** BGT.R -5
- **Significado:** Salto condicional para o endereço relativo, nesse caso é o (-5).

AND (AND OPERATION BETWEEN TWO REGISTERS)

- **Exemplo em Assembly:** AND R1, R2
- **Significado:** Realiza uma operação logica entre os registradores.

OR (OR OPERATION BETWEEN TWO REGISTERS)

- **Exemplo em Assembly:** OR R1, R2
- **Significado:** Realiza uma operação logica entre os registradores.

XOR (XOR OPERATION BETWEEN TWO REGISTERS)

- **Exemplo em Assembly:** XOR R1, R2
- **Significado:** Realiza uma operação logica entre os registradores.

NOT (NOT OPERATION IN REGISTER)

- **Exemplo em Assembly:** NOT R2, R3
- **Significado:** Realiza uma operação logica NOT no segundo registrador (R3), e armazena o resultado no registrador destino (R2).

END (END PROGRAM)

- **Exemplo em Assembly:** END
- **Significado:** Finaliza a execução do processador por completo.

IN (INPUT)

- **Exemplo em Assembly:** IN
- **Significado:** Captura o INPUT do teclado do usuário com base em endereços mapeados na memória RAM.

DRAW (DRAW CHAR)

- **Exemplo em Assembly:** DRAW
- **Significado:** Desenha na tela um caractere ASCII com base no endereço mapeado na memória RAM.

OUT (OUTPUT)

- **Exemplo em Assembly:** OUT
- **Significado:** Exibe no OUTPUT um número inteiro com base no endereço mapeado na memória RAM.

CALL (CALL SUB-ROUTINE)

- **Exemplo em Assembly:** CALL &34 | CALL SOMA *("Label")*
- **Significado:** Chama uma sub-rotina no endereço especificado ou chama uma label especificada.

RET (RETURN SUB-ROUTINE)

- **Exemplo em Assembly:** RET
- **Significado:** Retorna para próxima instrução do programa principal salvo no endereço da pilha SP.

INI.P (INITIALIZE POINTER)

- **Exemplo em Assembly:** INI.P &49152
- **Significado:** Inicializa o Global Pointer (GP), apontando para algum endereço de memória. Por padrão o GP aponta para o endereço do início da RAM (0xC000, ou seja, 49.152).

SET.P (SET POINTER)

- **Exemplo em Assembly:** SET.P R1
- **Significado:** Seta/Insere um valor de 8 bits ao endereço de memória apontado em GP.

GET.P (GET POINTER)

- **Exemplo em Assembly:** GET.P R2
- **Significado:** Captura um valor de 8 bits do endereço de memória apontado em GP.

UPD.P (UPDATE POINTER)

- **Exemplo em Assembly:** UPD.P R3
- **Significado:** Atualiza o valor do ponteiro GP, pegando seu endereço apontado e somando ou subtraindo-o, *dependendo do sinal e do valor contido no registrador*.

UPI.P (UPDATE IMMEDIATE POINTER)

- **Exemplo em Assembly:** UPI.P #-1
- **Significado:** Atualiza o valor do ponteiro GP, pegando seu endereço apontado e somando ou subtraindo-o, *dependendo do sinal e do valor imediato que foi passado*.

GETC (GET CHAR INPUT)

- **Exemplo em Assembly:** GETC
- **Significado:** Captura o INPUT do teclado do usuário em formato de caractere ASCII, usando os endereços mapeados na memória RAM.