INSTRUÇÕES

**CARREGA CONSTANTE NO BYTE MENOS SIGNIFICATIVO** **lcl c, Const8**

**Formato: lcl rc, offset8**

**Exemplo: lcl r3, offset8**

**Operação: *rc = Offset8 | (rc & 0xFF00)***

**Descrição: Carrega no byte menos significativo de *rc* o conteúdo de Offset 8.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CARREGA CONSTANTE NO BYTE MAIS SIGNIFICATIVO     lch c, Const8**

**Formato: lch rc, offset8**

**Exemplo: lch r3, offset8**

**Operação: *rc = (offset8 << 8) | (rc & 0x00ff)***

**Descrição: Carrega no byte mais significativo de *rc* o conteúdo de *Offset8*.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**DECREMENTO                                                                    deca rc,ra**

**Formato: deca rc, ra**

**Exemplo: deca r3, r1**

**Operação: *rc = ra - 1***

**Descrição: Subtrai 1 do conteúdo de *ra*e coloca o resultado em *rc*.**

**Flags afetados: neg, zero, carry, overflow.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**STORE WORD                                                                                store ra, rb**

**Formato: store ra, rb**

**Exemplo: store r3, r6**

**Operação: *memória [ra] = rb***

**Descrição: Carrega na posição da memória endereçada pelo registrador *a* o conteúdo do registrador *b*.**

**Flags afetados: nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**INCREMENTO                                                                inca rc,ra**

**Formato: inca rc, ra**

**Exemplo: inca r3, r1**

**Operação: *rc = ra + 1***

**Descrição: Adiciona 1 ao conteúdo de *ra*e coloca o resultado em *rc*.**

**Flags afetados: neg, zero, carry, overflow.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**COPIA *ra*   passa rc,ra**

**Formato: passa rc, ra**

**Exemplo: passa r4, r3**

**Operação: *rc = ra***

**Descrição: Faz conteúdo de *rc* igual ao conteúdo de *ra*.**

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**JUMP TRUE                                                                                jt.cond Destino**

**Formato: jt.cond Destino**

**Exemplo: jt.true loop**

**Operação: *se condição é verdadeira, PC = (PC + 1) + Offset8EstendidoPara16Bits***

**Descrição: Realiza desvio condicional relativo ao PC se a condição é verdadeira. O endereço**

**da próxima instrução a ser executada será o novo PC (PC + 1) mais a constante de 8** **bits**

**(presente nos bits de 0 a 7 da instrução) estendida de sinal.**

**Flags afetados: Nenhum.**

|  |  |
| --- | --- |
| **Condição** | **Mnemônico** |
| Resultado ALU negativo | .neg |
| Resultado ALU zero | .zero |
| Carry da ALU | .carry |
| Resultado ALU negativo ou zero | .negzero |
| Resultado ALU diferente de zero | .true |
| Resultado ALU *overflow* | .overflow |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**COPIA *ra*   passb rc,ra**

**Formato: passa rc, ra**

**Exemplo: passa r4, r3**

**Operação: *ra = rc***

**Descrição: Faz conteúdo de *ra* igual ao conteúdo de *rc*.**

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**zeros rc**

**Descrição: torna o valor de rc igual a zero**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**JUMP FALSE                                                                    jf.cond Destino**

**Tipo: IV**

**Formato: jf.cond offset8**

**Exemplo: jf.zero 10**

**Operação: *se condição é falsa, PC = (PC + 1) + Offset8EstendidoPara16Bits***

**Descrição: Realiza desvio condicional relativo ao PC se a condição é falsa. O endereço**

**da próxima instrução a ser executada será o novo PC (PC + 1) mais a constante de 8** **bits**

**(presente nos bits de 0 a 7 da instrução) estendida de sinal.**

**Flags afetados: Nenhum.**

|  |  |
| --- | --- |
| **Condição** | **Mnemônico** |
| Resultado ALU negativo | .neg |
| Resultado ALU zero | .zero |
| Carry da ALU | .carry |
| Resultado ALU negativo ou zero | .negzero |
| Resultado ALU diferente de zero | .true |
| Resultado ALU *overflow* | .overflow |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SHIFT ARITMÉTICO PARA A DIREITA                                asr rc,ra**

**Formato: asr rc, ra**

**Exemplo: asr r3, r2**

**Operação: *rc = ra >> 1***

**Descrição: Coloca cada bit *rai* em *rci-1* (exceto *ra0*) e preenche com *ra15* a posição *rc15.***

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUBTRAÇÃO INTEIRA COM DECREMENTO                        subdec rc,ra,rb**

**Formato: subdec rc, ra, rb**

**Exemplo: subdec r3, r2, r1**

**Operação: *rc = ra - rb - 1***

**Descrição: Subtrai o conteúdo de *rb* do conteúdo de *ra* e subtrai 1 da diferença, colocando o** **resultado em *rc*.**

**Flags afetados: neg, zero, carry e overflow.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NOT *ra*passnota rc,ra**

**Tipo: I**

**Formato: passnota rc, ra**

**Exemplo: passnota r4, r3**

**Operação: *rc = ~ra***

**Descrição: Faz conteúdo de *rc* valer o complemento do conteúdo de *ra*.**

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**JUMP INCONDICIONAL                                                    j Destino**

**Formato: j Destino**

**Exemplo: j loop**

**Operação: *PC = (PC + 1) + Offset12EstendidoPara16Bits***

**Descrição: Realiza desvio de fluxo incondicional relativo ao PC. O endereço da próxima** **instrução a ser executada**

**será o novo PC (PC + 1) mais a constante de** **12** **bits (presente nos bits de 0 a 11 da instrução) estendida de sinal.**

**Flags afetados: Nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CARREGA CONSTANTE DE 11 BITS COM SINAL        loadlit c, Const**

**Tipo: II**

**Formato: loadlit rc, offset11**

**Exemplo: loadlit r3, 1000**

**Operação: *rc = Offset11EstendidoPara16Bits***

**Descrição: Carrega em *rc* o conteúdo de *Offset*11 estendido para 32 bits.**

**Flags afetados: Nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**JUMP AND LINK                                                                    jal rb**

**Formato: jal rb**

**Exemplo: jal r4**

**Operação: *r7 = (PC + 1) e PC  = rb***

**Descrição: Realiza chamadas a procedimentos guardando o endereço do PC (PC +1) no registrador** ***r7***

**(para o retorno após o procedimento) e colocando em PC o valor de *rb* (primeira instrução do procedimento).**

**Flags afetados: Nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**NOT *ra*e AND LÓGICO                                                        andnota rc,ra,rb**

**Formato: andnota rc, ra, rb**

**Exemplo: andnota r3, r2, r7**

**Operação: *rc = (~ra) & rb***

**Descrição: Efetua *and* lógico bit a bit de *ra*(negado bit a bit) com *rb* e coloca o resultado em *rc*.**

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SUBTRAÇÃO INTEIRA                                                                sub rc,ra,rb**

**Formato: sub rc, ra, rb**

**Exemplo: sub r3, r2, r1**

**Operação: *rc = ra - rb***

**Descrição: Subtrai o conteúdo de *rb* do conteúdo de *ra* e coloca o resultado em *rc*.**

**Flags afetados: neg, zero, carry, overflow.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AND LÓGICO                                                                                   and rc,ra,rb**

**Tipo: I**

**Formato: and rc, ra, rb**

**Exemplo: and r3, r2, r7**

**Operação: *rc = ra & rb***

**Descrição: Efetua and lógico bit a bit de *ra* com *rb* e coloca o resultado em *rc*.**

**Flags afetados: neg, zero. carry = 0. overflow = 0.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**SOMA INTEIRA                                                                               add rc,ra,r**

**Tipo: I**

**Formato: add rc, ra, rb**

**Exemplo: add r3, r2, r1**

**Operação: *rc = ra + rb***

**Descrição: Soma (aritmética) o conteúdo de *ra* com *rb* e coloca o resultado em *rc*.**

**Flags afetados: neg, zero, carry, overflow.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**JUMP REGISTER                                                                    jr rb**

**Tipo: VI**

**Formato: jr rb**

**Exemplo: jr r7**

**Operação:*PC = r7***

**Descrição: Armazena o conteúdo do registrador *b* em PC.**

**Flags afetados: Nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**LOAD WORD                                                                                    load rc, ra**

**Tipo: VII**

**Formato: load rc, ra**

**Exemplo: load r3, r6**

**Operação: *rc = memória [ra]***

**Descrição: Carrega no registrador *c* o conteúdo da memória endereçada pelo registrador *a*.**

**Flags afetados: nenhum.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

* A diretiva LOWBYTE **x** indica para o montador o byte menos significativo da constante ou rótulo **x**(lembre-se que todo rótulo está associado a um valor). HIGHBYTE **x**, de maneira similar, indica o byte mais significativo da constante ou rótulo **x**.
* Diversas instruções podem gerar NOPs. No caso, foi escolhida a condição jf.true 0x00, ou seja, uma instrução onde todos os 32 bits estão zerados.
* A instrução HALT (parada do sistema) deve ser implementada como **L: j L**, isto é, desvio incondicional para o mesmo endereço do desvio. O código da instrução de HALT é *0x2fff*.
* Qualquer segmento de programa deve ser iniciado pela diretiva **.pseg**, enquanto todo segmento de dados é iniciado através da diretiva **.dseg.**
* Início do arquivo: indicado pela linha **.module x**
* Fim do arquivo: indicado pela linha **.end**