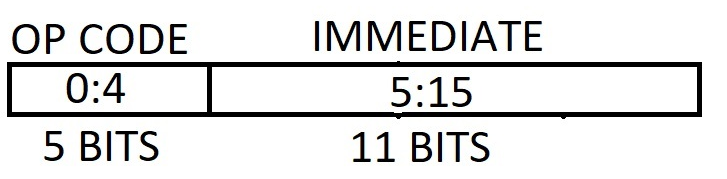
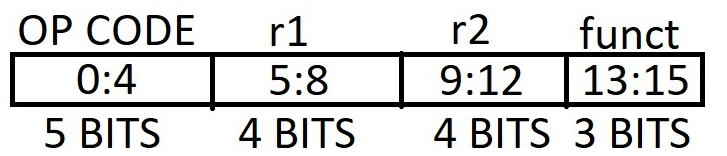


De modo geral, foram definidos dois tipos de instruções:  
  
- Uma instrução utilizando o imediato, no qual é definida pelo **opcode (0:4, unsigned) e um valor imediato (immediate (5:15, unsigned)) que definirão instruções como LI1, LI2, JUMPL, NOP e BEQ,** com estrutura exemplificada abaixo:  
 Seu uso consiste em analisar o opcode, identificando do que se trata a instrução, após isso, irá utilizar o imediato para um fim definido por cada operação.

- Outra instrução utilizando registrador, definida pelo **opcode (0:4, unsigned), com dois valores para registradores/endereços (r1 (5:8, unsigned) e r2 (9:12, unsigned) e funct (13:15, unsigned)) que definirão instruções como LOP1, LOP2, ADD, SUB, ADD, SLT, LW, SW, JUMPR, JLINK, entre outras instruções,** com estrutura exemplificada abaixo:

Seu uso consiste em analisar o opcode, identificando do que se trata a instrução, após isso, irá realizar operações, carregamentos ou buscas utilizando os registradores r1 e r2 e utilizando o campo funct para tomadas de decisão ou offset.

Abaixo, teremos os mnemônicos das instruções e exemplos de suas utilizações.

| Mnemônico | **SYSCALL** |
| --- | --- |
| Mapa de campos da sua instrução |  |
| *Opcode* | 00000 |
| Tamanho da instrução | 16bits |
| Quantidade de operandos | Zero operandos |
| Exemplo de uso da instrução | Depois de carregar nos registradores correspondentes o dado a ser manipulado e também a operação a ser feita, a chamada da instrução é  SYSCALL |
| Discriminação de cada operando (tamanho e tipo) | Sem operandos. IMMEDIATE não é usado |
| Tipo de operações que a instrução demanda | Interações com o sistema podem demandar diversas operações como ler dados da entrada ou escrever em algum periférico de saída. |
| Tipo de endereçamento da instrução | Não possui endereçamento |

| Mnemônico | **LI1 / LI2** | **NOP** | **BEQ/JUMPI** |
| --- | --- | --- | --- |
| Mapa de campos da sua instrução |  | | |
| *Opcode* | 01000/01001 | 01111 | 00011/00101 |
| Tamanho da instrução | 16bits | 16bits | 16bits |
| Quantidade de operandos | 1 | 0 | 1 |
| Exemplo de uso da instrução | LI1 -93;  LI2 123; | NOP; | BEQ 123;  JUMPI -123; |
| Discriminação de cada operando (tamanho e tipo) | **opcode (0:4, unsigned) , immediate (5:15, signed)**  Carrega nos registradores especiais de operando um imediato contido na instrução. | **opcode (0:4, unsigned) , immediate (5:15, signed)** - inutilizado  Instrução vazia, não realiza nada. | **opcode (0:4, unsigned) , immediate (5:15, signed)**  BEQ - se verdadeiro desloca o PC em N posições  JUMPI - desloca PC incondicionalmente em N posições |
| Tipo de operações que a instrução demanda | Carregamento no banco de registradores | nenhum | Leitura PC, Soma ao PC, escrita em PC |
| Tipo de endereçamento da instrução | - | - | Relativo ao PC : PC + IMM (IMM do tipo signed) |

| Mnemônico | **LOP1/LOP2/LOP12** | **ADD/ SUB** | **AND/OR/NOT** |
| --- | --- | --- | --- |
| Mapa de campos da sua instrução |  |  |  |
| *Opcode* | 01010 / 01011/ 01100 | 00001 / 10011 | 00010 / 10001 / 10101 |
| Tamanho da instrução | 16bits | 16bits | 16bits |
| Quantidade de operandos | 1 ou 2 operandos | 1 operando | 1 operando |
| Exemplo de uso da instrução | LOP1 x8  LOP2 x9  LOP12 x8, x9 | ADD x8  SUB x9 | AND x8  OR x9  NOT x3 |
| Discriminação de cada operando (tamanho e tipo) | LOP1 - o operando é **r1 (5:8, unsigned)** (r2 é desconsiderado), o endereço do registrador geral que será guardado no registrador especial de operando 1.  LOP2 - o operando é **r2(9:12, unsigned)** (r1 é desconsiderado).  LOP12 - os operandos são **r1(5:8, unsigned)** e r2(9:12, unsigned), os endereços dos registradores gerais que serão guardados, respectivamente, nos registrador especiais de operando 1 e 2. | O operando único **rd** **(5:8, unsigned)** (HI/REST é desconsiderado) é o registrador de destino da operação a ser realizada. Os operandos que serão somados ou subtraídos foram carregados previamente nos registradores especiais com LOPx. | O operando único **rd** **(5:8, unsigned)** (HI/REST é desconsiderado) é o registrador de destino da operação a ser realizada. Os operandos que serão usados foram carregados previamente nos registradores especiais com LOPx. |
| Tipo de operações que a instrução demanda | Leitura e escrita no banco de registradores | Operações de soma e subtração  Leitura e escrita no banco de registradores | Operações lógicas AND , OR e NOT  Leitura e escrita no banco de registradores |
| Tipo de endereçamento da instrução | Não possui endereçamento | Não possui endereçamento | Não possui endereçamento |

| Mnemônico | **MUL/DIV** | **SLT** | **EQ** |
| --- | --- | --- | --- |
| Mapa de campos da sua instrução |  |  |  |
| *Opcode* | 01110/00100 | 10010 | 10110 |
| Tamanho da instrução | 16bits | 16bits | 16bits |
| Quantidade de operandos | 1 operando | 1 operando | 1 operando |
| Exemplo de uso da instrução | MUL x8  DIV x9 | SLT x8 | EQ x3 |
| Discriminação de cada operando (tamanho e tipo) | Operandos **rd** **(5:8, unsigned)** e **HI/REST (9:12, unsigned)** são os registradores de destino da operação a ser realizada. Os operandos que serão multiplicados ou dividos foram carregados previamente nos registradores especiais com LOPx.  MUL - HI/REST (opcional) é o registrador com a parte mais significativa (HI) do resultado da multiplicação  DIV - HI/REST (opcional) é o registrador com o resto (REST) da divisão | O operando único **rd** **(5:8, unsigned)** (HI/REST é desconsiderado) é o registrador de destino da operação a ser realizada. Os operandos que serão comparados foram carregados previamente nos registradores especiais com LOPx. | O operando único **rd** **(5:8, unsigned)** (HI/REST é desconsiderado) é o registrador de destino da operação a ser realizada. Os operandos que serão comparados foram carregados previamente nos registradores especiais com LOPx. |
| Tipo de operações que a instrução demanda | Multiplicação, Divisão  Leitura e escrita em banco de registradores | Comparação  Leitura e escrita no banco de registradores | Comparação de igualdade  Leitura e escrita no banco de registradores |
| Tipo de endereçamento da instrução | Não possui endereçamento | Não possui endereçamento | Não possui endereçamento |

| Mnemônico | **JUMPL** | **JUMPR** |
| --- | --- | --- |
| Mapa de campos da sua instrução |  | |
| *Opcode* | 00110 | 00111 |
| Tamanho da instrução | 16 bits | 16 bits |
| Quantidade de operandos | 1 | 1 |
| Exemplo de uso da instrução | JLINK x8; | JUMPR xRa; |
| Discriminação de cada operando (tamanho e tipo) | **opcode (0:4, unsigned)** e **raddr (5:8, unsigned)** Utiliza os 13 bits menos significativos para obter o salto (visto que a memória necessita somente de 13 bits para navegá-la inteiramente) e armazena o endereço de retorno em **raddr**.  O campo funct **(13:15, unsigned)** definirá o offset relativo ao endereço raddr, sendo este positivo e limitado de 0-7 deslocamentos visto a restrição de 3 bits para o campo funct. | **opcode (0:4, unsigned)** e **raddr (5:8, unsigned)** Utiliza os 13 bits menos significativos para obter o salto (visto que a memória necessita somente de 13 bits para navegá-la inteiramente).  O campo funct **(13:15, unsigned)** definirá o offset relativo ao endereço raddr, sendo este positivo e limitado de 0-7 deslocamentos visto a restrição de 3 bits para o campo funct. |
| Tipo de operações que a instrução demanda | Leitura do banco de registradores, soma, escrita no PC, escrita no banco de registradores. | Leitura do banco de registradores, soma, escrita no PC. |
| Tipo de endereçamento da instrução | Registrador (com deslocamento) | Registrador (com deslocamento) |

| Mnemônico | **LW** | **SW** |
| --- | --- | --- |
| Mapa de campos da sua instrução |  | |
| *Opcode* | 01101 | 10100 |
| Tamanho da instrução | 16 bits | 16 bits |
| Quantidade de operandos | 2 | 2 |
| Exemplo de uso da instrução | LW x1, x5 (2) | SW x1, x5 (2) |
| Discriminação de cada operando (tamanho e tipo) | Carrega em rdata **(5:8, unsigned)** o conteúdo que estava na memória no endereço contido em raddr **(9:12, unsigned)**  O campo funct **(13:15, unsigned)** definirá o offset relativo ao endereço raddr, sendo este positivo e devido aos bits disponíveis para o mesmo, limitado a 0-7 deslocamentos de endereço. | Guarda o conteúdo de rdata **(5:8, unsigned)** na memória no endereço contido em raddr **(9:12, unsigned).**  O campo funct **(13:15, unsigned)** definirá o offset relativo ao endereço raddr, sendo este positivo e devido aos bits disponíveis para o mesmo, limitado a 0-7 deslocamentos de endereço. |
| Tipo de operações que a instrução demanda | Leitura na memória, leitura banco de registradores, soma, carregamento de banco de registradores | Escrita na memória, leitura banco de registradores, soma, carregamento de banco de registradores |
| Tipo de endereçamento da instrução | Registrador (com deslocamento) | Registrador (com deslocamento) |

| Mnemônico | **MVEPC** | **MVCAUSE** |
| --- | --- | --- |
| Mapa de campos da sua instrução |  | |
| *Opcode* | 10111 | 11000 |
| Tamanho da instrução | 16 bits | 16 bits |
| Quantidade de operandos | 1 | 1 |
| Exemplo de uso da instrução | MVEPC $a0 | MVCAUSE $a0 |
| Discriminação de cada operando (tamanho e tipo) | Carrega em rdata **(5:8, unsigned)** o conteúdo do registrador EPC, ou seja, o endereço da instrução a qual ocorreu a exceção Demais campos inutilizados | Carrega em rdata **(5:8, unsigned)** o conteúdo do registrador CAUSE, ou seja, o código da exceção ocorrida.  Demais campos inutilizados |
| Tipo de operações que a instrução demanda | Escrita no banco de registradores | Escrita no banco de registradores |
| Tipo de endereçamento da instrução | Registrador | Registrador |

| Número | Nome | Função |
| --- | --- | --- |
| 0 | $0 | Sempre zero |
| 1 | $v0 | Retorno função |
| 2 | $a0 | Argumento 0 |
| 3 | $a1 | Argumento 1 |
| 4 | $t0 | Temporário 0 |
| 5 | $t1 | Temporário 1 |
| 6 | $t2 | Temporário 2 |
| 7 | $t3 | Temporário 3 |
| 8 | $t4 | Temporário 4 |
| 9 | $s0 | Stored 0 |
| 10 | $s1 | Stored 1 |
| 11 | $s2 | Stored 2 |
| 12 | $s3 | Stored 3 |
| 13 | $s4 | Stored 4 |
| 14 | $sp | Stack pointer |
| 15 | $ra | Return Address |