

Usuário: User

1 / 1

Sobre o processador z80, existe a instrução de multiplicação?

Data: Domingo, 25 de junho de 2023 às 23:22:14 - (America/Sao_Paulo)

URL: <https://chat.openai.com/c/2e655d4a-4549-4cb0-bea9-0b993667990e>

Sim, o processador Z80 possui uma instrução de multiplicação embutida. A instrução de multiplicação no Z80 é chamada de "MUL" e é usada para multiplicar dois números de 8 bits (um byte) e produzir um resultado de 16 bits (dois bytes). A sintaxe da instrução MUL é a seguinte:

MUL registro

Onde "registro" é um registrador de 8 bits que contém o multiplicador. O multiplicando é considerado armazenado no par de registradores A e B. O resultado da multiplicação é armazenado nos pares de registradores A e B (parte alta) e C e D (parte baixa).

A instrução MUL realiza a multiplicação sem sinal. Portanto, se você desejar fazer uma multiplicação com números com sinal, precisará realizar o tratamento adequado dos sinais antes e depois da multiplicação.

Vale ressaltar que o processador Z80 é um processador antigo, lançado pela Zilog na década de 1970. Embora tenha sido amplamente utilizado em muitos sistemas computacionais e dispositivos eletrônicos da época, hoje em dia é considerado um processador obsoleto.

P 1:

Qual é a instrução em binário do MUL?

A instrução MUL no processador Z80 é codificada como 8 bits em binário. A codificação binária da instrução MUL é:

"11110110"

Essa sequência de bits representa a instrução MUL. Quando essa sequência de bits é encontrada na memória e interpretada pelo processador Z80, ele executa a instrução de multiplicação.

P 2:

existe a instrução de divisão no z80?

O processador Z80 não possui uma instrução de divisão embutida. Ao contrário da instrução de multiplicação (MUL), o Z80 não possui uma instrução específica para a divisão de números.

No entanto, é possível implementar a divisão usando uma sequência de instruções aritméticas e de controle de fluxo disponíveis no conjunto de instruções do Z80. Geralmente, a divisão é realizada por meio de iterações, subtraindo o divisor do dividendo repetidamente até que o resultado seja alcançado.

A implementação da divisão no Z80 pode ser complexa e requer manipulação cuidadosa dos registros e controle de fluxo. Normalmente, um programa de software especializado seria desenvolvido para realizar a divisão usando instruções aritméticas e de controle de fluxo disponíveis no Z80.

É importante observar que o Z80 é um processador antigo e existem processadores mais

modernos e eficientes disponíveis atualmente, que possuem instruções de multiplicação e divisão incorporadas em seu conjunto de instruções.

P 3:

Qual é o a instrução em binario do ADD?

A instrução ADD no processador Z80 é usada para adicionar dois valores e é codificada como 8 bits em binário. A codificação binária da instrução ADD é a seguinte:

"10000rrr"

Nessa sequência de bits, os três bits "rrr" representam o registrador de destino onde o valor será adicionado. Existem oito registradores de propósito geral no Z80, identificados como: B (000), C (001), D (010), E (011), H (100), L (101), A (111). Portanto, a instrução ADD pode ser seguida por qualquer um desses códigos binários para especificar o registrador de destino.

Por exemplo, para adicionar o valor do registrador B ao registrador A, a instrução ADD seria codificada como:

"10000000"

Esse é apenas um exemplo. A instrução ADD pode ser usada para adicionar valores entre diferentes registradores e também entre valores imediatos e registradores. A codificação binária completa da instrução ADD pode variar dependendo dos operandos e modos de endereçamento específicos usados.

P 4:

Qual a chamada de systema para imprimir?

No contexto do processador Z80, não há uma chamada de sistema específica para impressão. O Z80 é um processador de arquitetura de 8 bits que não possui um sistema operacional embutido. Portanto, não há chamadas de sistema padrão para operações de entrada e saída.

No entanto, ao utilizar o Z80 em conjunto com um sistema ou ambiente específico, é possível usar instruções de E/S (entrada/saída) específicas desse sistema para realizar operações de impressão. A maneira exata de realizar uma operação de impressão depende do sistema operacional ou ambiente no qual o Z80 está sendo usado.

Por exemplo, em um sistema que utiliza o CP/M (Control Program for Microcomputers), um sistema operacional popular para computadores baseados no Z80, a chamada de sistema para impressão seria feita por meio da interrupção do sistema com o número apropriado do serviço de impressão.

Em resumo, a chamada de sistema específica para impressão no Z80 varia de acordo com o sistema operacional ou ambiente específico no qual o processador está sendo usado. É necessário consultar a documentação ou as especificações do sistema em questão para obter informações detalhadas sobre a chamada de sistema correta para impressão.

P 5:

Qual a chamada de systema para sair?

A chamada de sistema para sair ou encerrar um programa pode variar dependendo do sistema

operacional ou ambiente em que o processador Z80 está sendo executado.

Em muitos sistemas que utilizam o CP/M (Control Program for Microcomputers), um sistema operacional popular para computadores baseados no Z80, a chamada de sistema para sair é feita por meio da interrupção do sistema com o número do serviço apropriado.

No CP/M, a chamada de sistema para encerrar um programa é feita através da interrupção 0x22. O registrador C deve ser configurado com o valor 0x00 antes de chamar essa interrupção.

Aqui está um exemplo em assembly Z80 mostrando como realizar a chamada de sistema para sair usando a interrupção 0x22 no CP/M:

assembly

```
MOV C, 0x00    ; Configura o registrador C com o valor 0x00
INT 0x22       ; Chama a interrupção do sistema para sair
```

Lembrando que essa é apenas uma forma de encerrar um programa em um sistema específico (CP/M) e pode variar em outros sistemas operacionais ou ambientes. É importante consultar a documentação ou as especificações do sistema em questão para obter informações precisas sobre a chamada de sistema correta para sair ou encerrar um programa.

P 6:

Qual é a instrução em binário para sair do sistema do z80?

No processador Z80, não há uma instrução específica em binário para "sair do sistema". O Z80 é um processador de arquitetura de 8 bits que não possui uma instrução nativa para finalizar ou encerrar um programa.

A funcionalidade de "sair do sistema" é normalmente implementada usando chamadas de sistema ou interrupções específicas do sistema operacional ou ambiente em que o Z80 está sendo utilizado.

Portanto, a instrução binária específica para "sair do sistema" no Z80 depende do sistema operacional ou ambiente específico em que o processador está sendo executado. Cada sistema operacional pode ter sua própria forma de realizar essa operação, por meio de interrupções ou chamadas de sistema específicas.

Para determinar a instrução binária exata para "sair do sistema" no seu contexto específico, é necessário consultar a documentação ou as especificações do sistema operacional ou ambiente em questão.

P 7:

Qual é o a instrução em binario AND no z80?

A instrução AND no processador Z80 é usada para realizar uma operação lógica "E" (AND) entre dois valores. A codificação binária da instrução AND é a seguinte:

"10100rrr"

Nessa sequência de bits, os três bits "rrr" representam o registrador de destino onde o valor resultante será armazenado. Existem oito registradores de propósito geral no Z80, identificados como: B (000), C (001), D (010), E (011), H (100), L (101), A (111). Portanto, a instrução AND pode ser seguida por qualquer um desses códigos binários para especificar o registrador de destino.

Por exemplo, para realizar a operação AND entre o valor do registrador B e o valor do registrador A, a instrução AND seria codificada como:

"10100000"

Essa instrução irá executar uma operação lógica "E" (AND) bit a bit entre o conteúdo dos registradores B e A, e o resultado será armazenado no registrador A.

É importante observar que a instrução AND no Z80 opera em nível de bits, aplicando a operação lógica "E" (AND) entre os bits correspondentes dos operandos.

P 8:

me mostre um exemplo para essa instrução

Claro! Aqui está um exemplo de código assembly Z80 que utiliza a instrução AND para realizar uma operação lógica "E" (AND) entre o conteúdo dos registradores A e B, armazenando o resultado no registrador A:

assembly

```
LD A, 0xFF      ; Carrega o valor 0xFF no registrador A

LD B, 0xAA      ; Carrega o valor 0xAA no registrador B


AND B           ; Realiza a operação AND entre A e B,
armazenando o resultado em A


; O registrador A agora contém o resultado da operação AND


HALT            ; Encerra a execução do programa
```

Nesse exemplo, inicialmente os valores 0xFF e 0xAA são carregados nos registradores A e B, respectivamente. Em seguida, a instrução AND é utilizada para realizar a operação lógica "E" (AND) bit a bit entre o conteúdo dos registradores A e B. O resultado da operação é armazenado no registrador A.

Após a execução da instrução AND, o registrador A conterá o resultado da operação lógica "E" (AND) entre os valores 0xFF e 0xAA. O programa, em seguida, pode executar outras instruções ou ser finalizado com uma instrução como o HALT.

Observe que os valores utilizados neste exemplo são apenas ilustrativos. Você pode substituí-los por outros valores para realizar a operação AND desejada entre os registradores no seu programa.

P 9:

me mostre outro exemplo

Com certeza! Aqui está outro exemplo de código assembly Z80 que utiliza a instrução AND para realizar uma operação lógica "E" (AND) entre o conteúdo de um registrador e um valor imediato:

assembly

```
LD A, 0x3C      ; Carrega o valor 0x3C no registrador A

AND 0x0F        ; Realiza a operação AND entre o valor imediato
0x0F e o conteúdo de A

; O registrador A agora contém o resultado da operação AND

HALT            ; Encerra a execução do programa
```

Neste exemplo, o valor 0x3C é carregado no registrador A. Em seguida, a instrução AND é usada para realizar a operação lógica "E" (AND) bit a bit entre o valor imediato 0x0F e o conteúdo do registrador A. O resultado da operação é armazenado novamente no registrador A.

Após a execução da instrução AND, o registrador A conterá o resultado da operação lógica "E" (AND) entre os bits correspondentes do valor imediato 0x0F e o valor originalmente presente no registrador A.

Você pode experimentar diferentes valores imediatos e registradores para realizar diferentes operações lógicas "E" (AND) no seu programa.

P 10:

Faça uma função em python que receba um valor inteiro e retorne o binário dele

Claro! Aqui está uma função em Python que recebe um valor inteiro e retorna sua representação binária como uma string:

python

```
def inteiro_para_binario(valor):  
    if valor == 0:  
        return '0' # Tratamento especial para o valor zero  
  
    binario = ''  
    while valor > 0:  
        bit = valor % 2  
        binario = str(bit) + binario  
        valor = valor // 2  
  
    return binario
```

A função "inteiro_para_binario" recebe um valor inteiro como argumento e, usando a divisão sucessiva por 2, calcula os dígitos binários correspondentes. Os dígitos binários são concatenados da direita para a esquerda, formando a representação binária completa.

Aqui está um exemplo de como usar essa função:

python

```
numero = 42  
binario = inteiro_para_binario(numero)  
print(binario) # Saída: '101010'
```

Neste exemplo, o valor inteiro 42 é convertido em sua representação binária, que é '101010'. Você pode chamar a função "inteiro_para_binario" com qualquer valor inteiro e ela retornará sua representação binária correspondente como uma string.

P 11:

me mostre um exemplo da instrução OR em em binario no z80

A instrução OR no processador Z80 é usada para realizar uma operação lógica "OU" (OR) entre dois valores. A codificação binária da instrução OR é a seguinte:

"10110rrr"

Nessa sequência de bits, os três bits "rrr" representam o registrador de destino onde o valor resultante será armazenado. Assim como nas outras instruções, existem oito registradores de propósito geral no Z80, identificados como: B (000), C (001), D (010), E (011), H (100), L (101),

A (111). Portanto, a instrução OR pode ser seguida por qualquer um desses códigos binários para especificar o registrador de destino.

Por exemplo, para realizar a operação OR entre o valor do registrador B e o valor do registrador A, a instrução OR seria codificada como:

"10110000"

Essa instrução irá executar uma operação lógica "OU" (OR) bit a bit entre o conteúdo dos registradores B e A, e o resultado será armazenado no registrador A.

A instrução OR pode ser usada para combinar os bits correspondentes de dois valores e produzir um resultado que tenha um bit definido sempre que pelo menos um dos bits correspondentes dos operandos seja definido.

É importante ressaltar que essa é apenas a representação binária da instrução OR no Z80. A forma completa da instrução pode variar dependendo dos operandos e modos de endereçamento específicos usados.

P 12:

me mostre um exemplo

Aqui está um exemplo de código assembly Z80 que utiliza a instrução OR para realizar uma operação lógica "OU" (OR) entre o conteúdo dos registradores A e B, armazenando o resultado no registrador A:

assembly

```
LD A, 0x0F      ; Carrega o valor 0x0F no registrador A
LD B, 0x3C      ; Carrega o valor 0x3C no registrador B

OR B            ; Realiza a operação OR entre A e B,
armazenando o resultado em A

; O registrador A agora contém o resultado da operação OR

HALT           ; Encerra a execução do programa
```

Nesse exemplo, inicialmente, os valores 0x0F e 0x3C são carregados nos registradores A e B, respectivamente. Em seguida, a instrução OR é utilizada para realizar a operação lógica "OU" (OR) bit a bit entre o conteúdo dos registradores A e B. O resultado da operação é armazenado no registrador A.

Após a execução da instrução OR, o registrador A conterá o resultado da operação lógica "OU" (OR) entre os valores 0x0F e 0x3C. O programa pode executar outras instruções ou ser finalizado com uma instrução como o HALT.

Lembre-se de que os valores utilizados neste exemplo são apenas ilustrativos. Você pode substituí-los por outros valores para realizar a operação OR desejada entre os registradores no seu programa.

P 13:

existe a instrução NOR no z80?

No processador Z80, não existe uma instrução NOR (ou "OR Negado") nativamente implementada. A instrução NOR é uma operação lógica que retorna o resultado "1" apenas se ambos os bits de entrada forem "0". Embora o Z80 possua instruções lógicas básicas, como AND e OR, não possui uma instrução específica para NOR.

No entanto, é possível obter o efeito da operação NOR por meio de combinações de instruções existentes. Uma maneira de realizar uma operação NOR é através de uma sequência de instruções que envolvem as instruções NOT (inversão lógica) e AND. A lógica NOR pode ser representada pela seguinte fórmula: $\text{NOT}(A) \text{ AND } \text{NOT}(B)$.

Aqui está um exemplo de como simular uma operação NOR no Z80 usando as instruções existentes:

assembly

```
    ; Supondo que A e B contêm os valores a serem usados na
    operação NOR

    CPL      ; Complementa o valor em A (inversão lógica)
    AND B    ; Realiza a operação AND entre o valor complementado
    em A e B

    ; O resultado estará no registrador A após essa sequência de
    instruções
```

Neste exemplo, o valor em A é complementado usando a instrução CPL (Complement) para obter o complemento lógico de A. Em seguida, é realizada uma operação AND entre o valor complementado em A e o valor em B. O resultado final estará no registrador A após essa sequência de instruções, representando o resultado da operação NOR entre os valores originais em A e B.

Lembre-se de que a ausência de uma instrução NOR nativa no Z80 requer a combinação de instruções existentes para simular a operação desejada.

P 14:

Qual é a instrução do z80 que é igual a do MIPS como SLT

A instrução no processador Z80 que é equivalente à instrução SLT (Set Less Than) do MIPS é a instrução CP (Compare). A instrução CP compara dois valores e define o bit de sinal (bit 7) do registrador de estado (flags) para indicar se o valor do primeiro operando é menor que o valor do segundo operando.

A codificação binária da instrução CP no Z80 é a seguinte:

"10111rrr"

Nessa sequência de bits, os três bits "rrr" representam o registrador que será usado como o primeiro operando na comparação.

Por exemplo, para realizar uma comparação entre o valor do registrador A e o valor do

registrador B, a instrução CP seria codificada como:

"10111000"

Após a execução da instrução CP, o bit de sinal do registrador de estado (flags) será definido com base na comparação realizada entre os valores dos registradores A e B.

É importante observar que a instrução CP apenas compara os valores e define o bit de sinal, mas não realiza nenhuma operação de armazenamento do resultado. O resultado da comparação pode ser utilizado posteriormente por meio de outras instruções condicionais, como JR (Jump Relative) ou JP (Jump Relative on Plus), que verificam o estado do bit de sinal para tomar decisões de salto condicional.

P 15:

A instrução no processador Z80 que é equivalente à instrução SLL do MIPS

O processador Z80 não possui uma instrução diretamente equivalente à instrução SLL (Shift Left Logical) do MIPS. A instrução SLL no MIPS realiza um deslocamento lógico para a esquerda dos bits de um registrador, preenchendo os bits de menor ordem com zeros.

No entanto, você pode obter um efeito semelhante no Z80 usando a instrução SLA (Shift Left Arithmetic) ou a instrução DAA (Decimal Adjust Accumulator) em certos casos específicos.

A instrução SLA realiza um deslocamento aritmético para a esquerda dos bits de um registrador, preenchendo o bit de sinal (bit mais significativo) com o valor do bit anterior de maior ordem. O resultado é armazenado no próprio registrador. Embora a instrução SLA não

seja exatamente igual à instrução SLL, em muitos casos ela pode ser usada para obter resultados semelhantes.

A instrução DAA é usada para ajustar o resultado de uma operação aritmética no formato BCD (Binary Coded Decimal). Embora não seja diretamente relacionada à operação de deslocamento lógico, em alguns casos específicos a instrução DAA pode ser utilizada para realizar um deslocamento lógico para a esquerda, resultando em efeitos semelhantes à instrução SLL.

No entanto, é importante destacar que as instruções SLA e DAA no Z80 possuem comportamentos e propósitos diferentes da instrução SLL do MIPS. Portanto, é necessário adaptar e ajustar seu código para obter resultados semelhantes aos desejados com a instrução SLL.

P 16:

Qual a instrução no processador Z80 que é equivalente à instrução SRL do MIPS

A instrução no processador Z80 que é equivalente à instrução SRL (Shift Right Logical) do MIPS é a instrução SRL (Shift Right Logical) também no Z80. Ambas as instruções realizam um deslocamento lógico para a direita dos bits de um registrador, preenchendo os bits de maior ordem com zeros.

A codificação binária da instrução SRL no Z80 é a seguinte:

"11001011"

Essa sequência de bits representa a instrução SRL. Após a instrução, você precisa especificar

o registrador no qual o deslocamento será realizado.

Por exemplo, para realizar um deslocamento lógico para a direita dos bits no registrador A, a instrução SRL seria codificada como:

"11001011 111"

Nesse caso, os três bits "111" no final da instrução representam o registrador A.

Após a execução da instrução SRL, os bits no registrador A serão deslocados para a direita, preenchendo os bits de maior ordem com zeros.

Lembre-se de que a instrução SRL realiza um deslocamento lógico, ou seja, não mantém o valor do bit de sinal (bit mais significativo) ao realizar o deslocamento. Se você precisar preservar o valor do bit de sinal, pode usar a instrução SRA (Shift Right Arithmetic), que realiza um deslocamento aritmético e mantém o valor do bit de sinal.

P 17:

Me mostre um exemplo em binário da instrução CP

Claro! Aqui está um exemplo da instrução CP (Compare) no processador Z80, na representação binária:

"10111rrr"

Nessa sequência de bits, os três bits "rrr" representam o registrador que será usado como o

primeiro operando na comparação. Existem oito registradores de propósito geral no Z80: B (000), C (001), D (010), E (011), H (100), L (101), A (111). Portanto, a instrução CP pode ser seguida por qualquer um desses códigos binários para especificar o registrador de destino.

Por exemplo, para realizar uma comparação entre o valor do registrador A e o valor do registrador B, a instrução CP seria codificada como:

"10111000"

Nesse exemplo, a instrução CP é seguida pelo código binário "000", que representa o registrador B. Essa sequência de bits resulta na comparação do valor do registrador A com o valor do registrador B.

A instrução CP compara os valores e define o bit de sinal (bit 7) do registrador de estado (flags) com base na comparação realizada. O resultado da comparação pode ser utilizado posteriormente por meio de outras instruções condicionais para tomar decisões de salto condicional ou outras operações lógicas.

P 18:

onde é armazenado este resultado?

Desculpe pelo equívoco anterior. A instrução CP no processador Z80 não armazena explicitamente o resultado da comparação em um registrador específico. Em vez disso, ela afeta o registrador de estado (flags) do processador.

O resultado da comparação é refletido nos bits de estado do registrador de flags, que são

chamados de Zero (Z), Sign (S), Carry (C) e Parity/Overflow (P/V), entre outros. Esses bits de estado são utilizados para indicar informações sobre o resultado de várias operações, incluindo a instrução CP.

Após a execução da instrução CP, os bits de estado do registrador de flags são atualizados com base no resultado da comparação. Por exemplo, se os valores comparados forem iguais, o bit de Zero (Z) será definido para indicar que o resultado é zero. Se o primeiro valor for menor que o segundo, o bit de Carry (C) será definido. O bit de Sign (S) será definido se o resultado for negativo, e o bit de Parity/Overflow (P/V) terá seu valor definido de acordo com regras específicas.

Você pode usar instruções condicionais, como JR (Jump Relative), JP (Jump Relative on Plus), JNZ (Jump if Not Zero), JZ (Jump if Zero), entre outras, para verificar os bits de estado do registrador de flags e tomar decisões de salto condicional com base no resultado da comparação.

Portanto, diferentemente de outras instruções que armazenam explicitamente o resultado em um registrador, a instrução CP no Z80 atualiza os bits de estado do registrador de flags para refletir o resultado da comparação realizada.

P 19:

O SLT do mips tem mesma função do CP no z80?

Peço desculpas por qualquer confusão causada anteriormente. Para esclarecer, a função do SLT (Set Less Than) no MIPS e do CP (Compare) no Z80 é semelhante, mas existem diferenças importantes entre eles.

O SLT no MIPS realiza uma comparação entre dois valores e define um registrador de destino com o valor "1" se o primeiro valor for menor que o segundo; caso contrário, o registrador de destino é definido como "0".

Por outro lado, o CP no Z80 também realiza uma comparação entre dois valores, mas não armazena explicitamente o resultado em um registrador específico. Em vez disso, ele atualiza os bits de estado do registrador de flags para refletir o resultado da comparação.

Enquanto o SLT do MIPS armazena o resultado em um registrador de propósito geral, o CP do Z80 atualiza os bits de estado do registrador de flags, como o bit de Zero (Z), Sign (S), Carry (C) e Parity/Overflow (P/V), para indicar informações sobre o resultado da comparação.

Portanto, embora o SLT e o CP compartilhem a função de realizar uma comparação entre valores, eles têm abordagens diferentes para lidar com o resultado dessa comparação e onde armazenam esse resultado.

P 20:

Qual a instrução no processador Z80 que é equivalente à instrução SRA do MIPS

A instrução no processador Z80 que é equivalente à instrução SRA (Shift Right Arithmetic) do MIPS é a instrução SRA (Shift Right Arithmetic) também no Z80. Ambas as instruções realizam um deslocamento aritmético para a direita dos bits de um registrador, preservando o valor do bit de sinal (bit mais significativo) durante o deslocamento.

A codificação binária da instrução SRA no Z80 é a seguinte:

"11001011"

Essa sequência de bits representa a instrução SRA. Após a instrução, você precisa especificar o registrador no qual o deslocamento será realizado.

Por exemplo, para realizar um deslocamento aritmético para a direita dos bits no registrador A, a instrução SRA seria codificada como:

"11001011 111"

Nesse exemplo, os três bits "111" no final da instrução representam o registrador A.

Após a execução da instrução SRA, os bits no registrador A serão deslocados para a direita, mantendo o valor do bit de sinal. O bit de sinal é duplicado ou estendido durante o deslocamento para manter o valor original.

Portanto, a instrução SRA é usada para realizar deslocamentos aritméticos para a direita, mantendo o valor do bit de sinal, o que a torna equivalente à instrução SRA do MIPS.

P 21:

Qual a instrução no processador Z80 que é equivalente à instrução BEQ do MIPS?

A instrução no processador Z80 que é equivalente à instrução BEQ (Branch if Equal) do MIPS é a instrução JR (Jump Relative) combinada com a verificação do bit de Zero (Z) do registrador de flags.

No Z80, a instrução JR realiza um salto incondicional para um endereço relativo, especificado como um deslocamento em relação ao valor atual do Program Counter (PC). Para realizar uma

operação semelhante à instrução BEQ do MIPS, é necessário combinar a instrução JR com a verificação do bit de Zero (Z) do registrador de flags.

A sequência de instruções para simular o comportamento da instrução BEQ no Z80 seria a seguinte:

1. Verificar o bit de Zero (Z) usando a instrução JP (Jump Relative on Plus) para pular o próximo comando se o bit de Zero não estiver definido:

CSS

"JP NZ, label"

2. Executar a instrução JR (Jump Relative) para realizar o salto incondicional para o endereço desejado caso o bit de Zero esteja definido:

CSS

"JR label"

Nesse exemplo, a instrução JP (Jump Relative on Plus) é usada para verificar o bit de Zero (Z) do registrador de flags. Se o bit de Zero não estiver definido, significa que os valores comparados não são iguais, então ocorre o salto para a label especificada. Caso contrário, a execução continua com a instrução JR, que realiza o salto incondicional para a label.

Dessa forma, essa combinação de instruções permite simular o comportamento da instrução BEQ do MIPS no processador Z80.

P 22:

Transforme nossa conversa em PDF