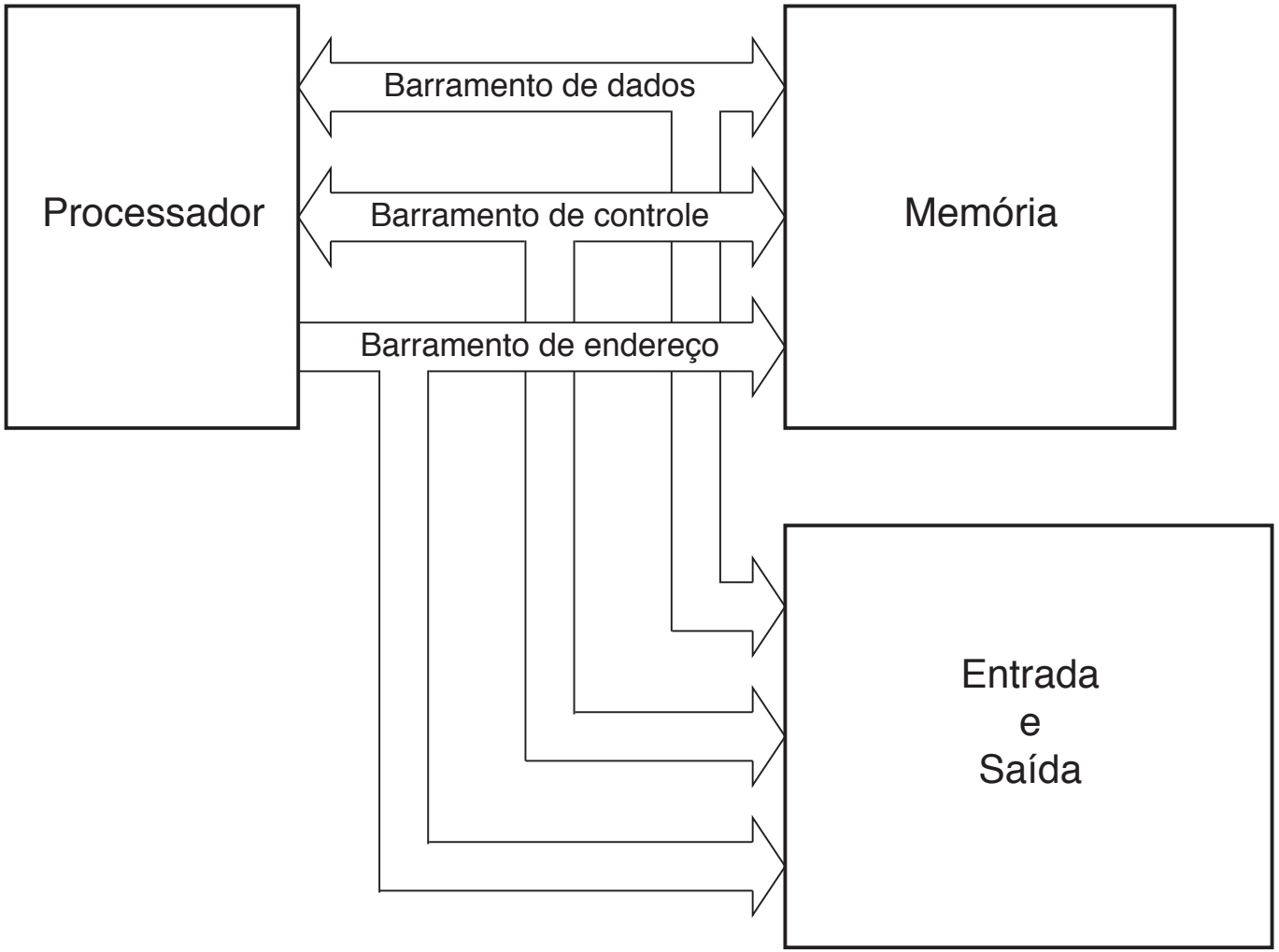
Linguagens de montagem
Capítulo 2 - Introdução à organização de computadores

Ricardo Anido

Esquema simplificado de um computador



Esquema simplificado de um computador

- O processador (CPU, *Central Processing Unity*) executa um conjunto fixo de instruções, e controla todos os outros componentes.
- Os *barramentos* são usados para transferir informações entre dois componentes. No momento da transferência, o componente que possui a informação coloca as tensões adequadas nos fios do barramento, e o componente destino faz a leitura das tensões nos fios, adquirindo assim a informação.

Barramentos

- São usados três barramentos: dados, endereço e controle.
- Nos barramentos de dados e de controle a informação pode trafegar nas duas direções, enquanto que o barramento de endereços a informação trafega em uma única direção.
- O barramento de endereços deve conter tantos fios (bits) quantos forem necessários para endereçar todas as posições da memória física instalada.
- O número de fios (bits) do barramento de dados é igual ao tamanho da palavra utilizada pelo processador.

Barramento de controle

Dois fios importantes no barramento de controle são:

- $rd/\overline{wr}$, que indica se o processador quer realizar uma operação de escrita ou de leitura (valor 1 significa leitura, 0 significa escrita); e
- $mem/\overline{io}$, que indica se a operação deve ser respondida pela memória ou pelo sistema de E/S (valor 1 significa acesso a memória, 0 significa acesso a E/S).

Comunicação entre processador e memória

Quando o processador necessita ler um dado na memória:

1. Especifica pelo barramento de endereços qual o endereço de memória da palavra que contém o dado.
2. Especifica pelo barramento de controle que a operação é de leitura em memória ($rd/\overline{wr} = 1$ e $mem/\overline{io} = 1$). [A memória então coloca o valor da palavra especificada no barramento de dados.]
3. Processador espera um tempo fixo pré-determinado para que a memória responda e que as tensões no barramentos de dados estejam estáveis (o tempo de espera é chamado de *tempo de preparação*, em inglês *setup time*).
4. Após o tempo de preparação, lê o valor fornecido pela memória no barramento de dados.

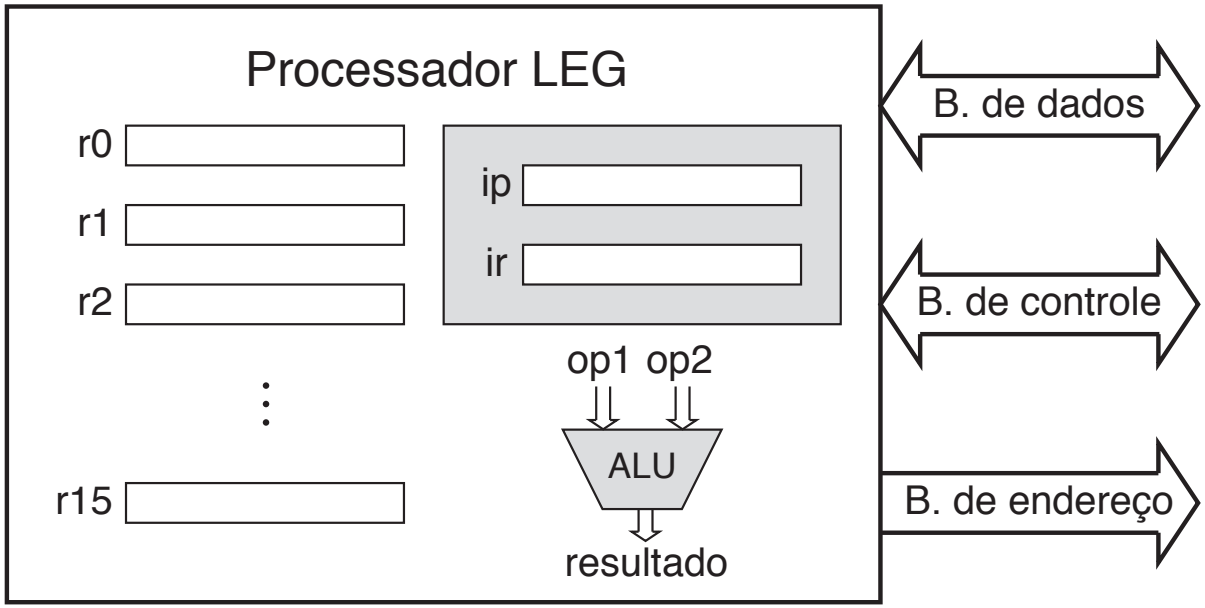
Comunicação entre processador e memória

A operação de escrita em memória é similar à leitura. O processador:

1. Coloca no barramento de endereços o endereço da palavra que deve ser modificada, e no barramento de dados o valor a ser escrito.
2. Indica pelo barramento de controle que a operação é de escrita em memória ($rd/\overline{wr} = 0$ e $mem/\overline{io} = 1$).
3. Espera o tempo de preparação especificado para que memória tenha tempo de executar a operação.

Esquema simplificado do processador LEG

Vamos examinar o funcionamento do processador através da introdução de um processador didático, o LEG.



Registradores

Os componentes mostrados no interior do processador (`r0`, `r1`, `r2`, ..., `r15`, `ip` e `ir`) são chamados *registradores*.

- Um registrador é uma palavra de memória interna ao processador, com acesso muito mais rápido do que o acesso a qualquer palavra de memória externa ao processador.
- O LEG possui vários registradores, todos com 32 bits.
- Os registradores `r0` a `r15` são de propósito geral; podem ser usados para manipular dados do usuário, armazenar valores intermediários, etc.

Registradores

São mostrados ainda dois registradores especiais:

- O registrador de instruções `ir` (em inglês, *instruction register*) armazena o código da instrução que está sendo executada.
- O registrador apontador de instruções `ip` (em inglês, *instruction pointer*) contém o endereço da próxima instrução a ser executada.
- O programador não tem acesso direto aos registradores especiais `ip` e `ir`; eles são mostrados nas figuras para facilitar o entendimento do funcionamento do processador.

ALU

- O processador conta ainda com uma Unidade Aritmética e Lógica (ALU, do nome em inglês *Arithmetic and Logic Unit*), que realiza operações aritméticas (por exemplo adição) e lógicas (por exemplo ou-exclusivo).
- NO LEG os operandos da ALU são registradores.

Funcionamento do processador

O processador funciona em passos.

- Cada instrução é composta por um número fixo de passos.
- Dependendo da complexidade da instrução alguns dos passos não têm qualquer atividade.

Funcionamento do processador

Os passos básicos de uma instrução são:

- busca de instrução (na memória),
- busca de operando (na memória ou E/S),
- execução e
- armazenamento do resultado (na memória ou E/S).

Passo Busca de Instrução (*fetch*)

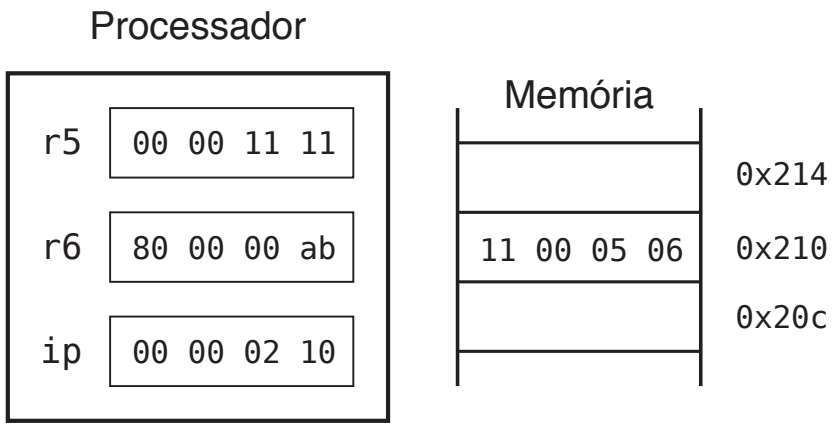
- O processador executa uma operação de leitura da memória, no endereço indicado pelo valor corrente do registrador apontador de instruções `ip`, para ler a instrução que deve ser executada.
- A palavra lida é colocada no registrador `ir`, e o valor do registrador `ip` é incrementado para apontar para a próxima palavra na memória.
- Se a instrução é composta por mais de uma palavra, um novo acesso à memória é realizado, o valor lido é armazenado internamente no processador, e o registrador `ip` é novamente incrementado para apontar para a próxima palavra da memória.

Outros passos

- No passo *busca de operando*, se necessário para a instrução, mais um acesso à memória ou à E/S é realizado, para buscar um operando para a instrução.
- No passo *execução* o processador utiliza a sua Unidade Lógica e Aritmética para realizar a operação especificada (por exemplo, adição ou subtração).
- Finalmente, no passo *armazenamento do resultado*, se necessário para a instrução, é feito um acesso de escrita à memória ou à E/S para armazenar o resultado da instrução.

Exemplo

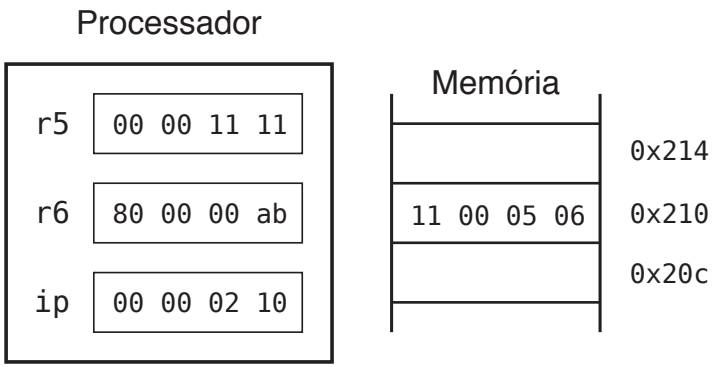
Suponha que o código `0x11000506` represente a instrução "adicione o valor do registrador `r5` ao valor do registrador `r6` e coloque o resultado em `r5`", e que em um dado momento a memória e os registradores contenham valores mostrados abaixo:



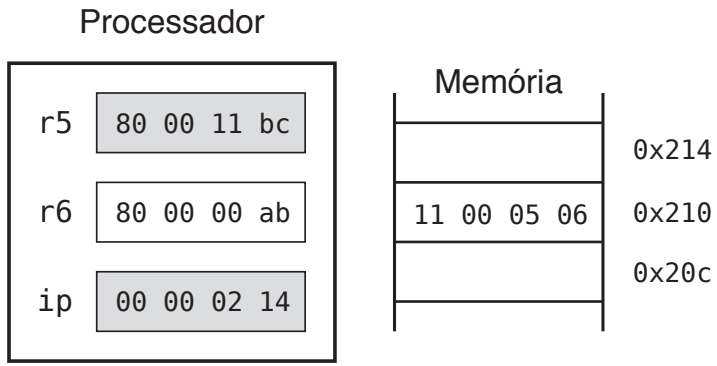
Exemplo

1. *Busca de instrução*: o processador lê a palavra de memória apontada por `ip` (endereço `0x0210`) e coloca o valor lido (`0x11000506`) no registrador interno `ir` (não mostrado na figura). O registrador `ip` é avançado para a próxima palavra: `ip` passa a valer `0x0214`.
2. *Busca de operando*: nada a fazer no caso desta instrução.
3. *Execução*: o processador executa a instrução correspondente ao código `0x11000506`: o processador aciona a Unidade Aritmética e Lógica tendo como entrada os valores dos registradores `r5` e `r6` e efetua a adição. O resultado é colocado no registrador `r5`.
4. *Armazenamento de resultado*: nada a fazer no caso desta instrução.

Exemplo



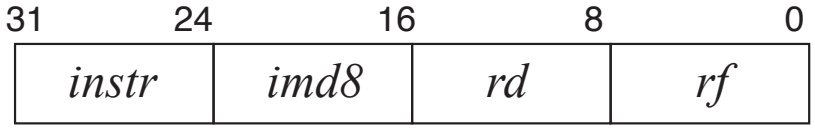
Estado antes da execução



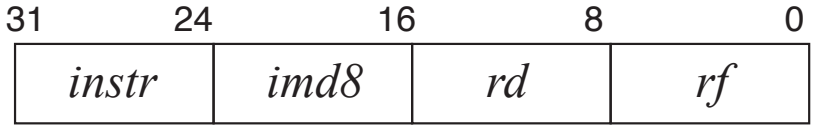
Estado após a execução

Codificação de instruções no LEG

A codificação de instruções no LEG é muito simples. Toda instrução tem uma ou duas palavras. A primeira palavra é sempre dividida em quatro campos de um byte cada: *instr*, *imd8*, *rd* e *rf*,



Codificação de instruções no LEG



- O campo *instr* representa o tipo da instrução.
- O campo *imd8* terá sua utilização explicada mais adiante.
- Os campos *rd* e *rf* codificam os registradores usados como *destino* e *fonte* na instrução. Os registradores são identificados nestes campos pelos seus números, ou seja, o registrador `r0` tem como representação o valor 0, e o registrador `r1` tem como representação o valor 1, e assim por diante.
- A utilização dos campos *rf* e *rd* dependem da instrução em questão; algumas utilizam apenas o campo *rd*, outras apenas o campo *rf* e algumas utilizam ambos os campos.

Codificação de instruções no LEG

Podemos agora entender a codificação da instrução "adicione o valor do registrador `r5` ao valor do registrador `r6` e coloque o resultado em `r5`", usada como exemplo anteriormente:

instr	*imd8*	*rd*	*rf*
`0x11`	`0x00`	`0x05`	`0x06`

operação: $rd = rf + rd$

Modos de endereçamento

- Instruções de leitura de memória copiam ("carregam") o valor de uma posição de memória em um registrador.
- Instruções de escrita em memória copiam o valor de um registrador em uma posição de memória.
- Em instruções que fazem acesso à memória é necessário indicar qual a palavra de memória que deve ser utilizada.

Modos de endereçamento

- Mais precisamente, é necessário especificar como deve ser calculado o endereço da palavra de memória a ser utilizada, chamado *endereço efetivo* do operando.
- Existem diversas formas possíveis de cálculo do endereço efetivo.
- Essas diferentes formas dão origem a diferentes modos (ou tipos) de endereçamento; cada instrução do processador define precisamente o modo de endereçamento utilizado.

Endereçamento imediato

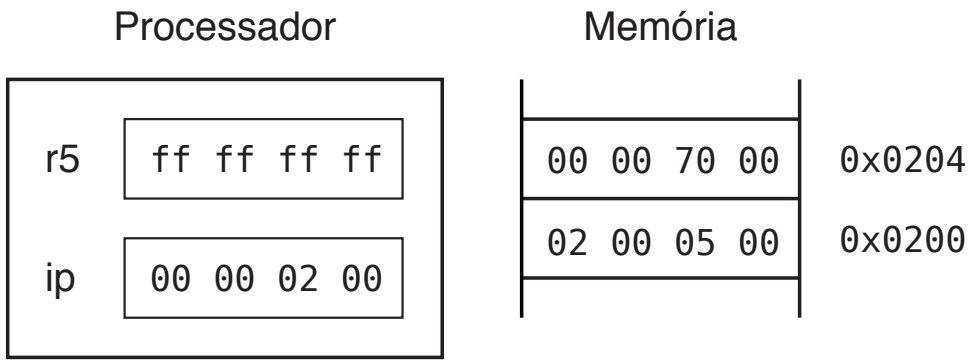
- Um dos modos mais simples de endereçamento é o chamado endereçamento imediato, no qual o valor do operando faz parte da codificação da instrução.
- O endereçamento imediato pode ser utilizado por exemplo para colocar em um registrador um valor constante.

Endereçamento imediato

- No LEG, a instrução que carrega uma constante em um registrador é "carrega registrador com valor constante", que é codificada utilizando-se duas palavras de memória.
- A primeira palavra especifica o tipo de operação (carrega registrador com endereçamento imediato) e o registrador destino (para o qual o valor constante deve ser copiado).
- A segunda palavra da instrução contém o valor que deve ser carregado no registrador destino.

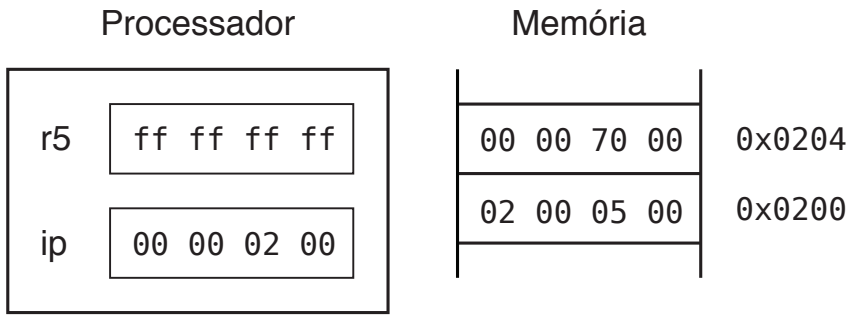
Exemplo

A Figura abaixo mostra a codificação da instrução "carrega registrador `r5` com o valor `0x7000`".



Exemplo

Suponha em antes da execução a memória e os registradores contenham valores mostrados abaixo:



Exemplo

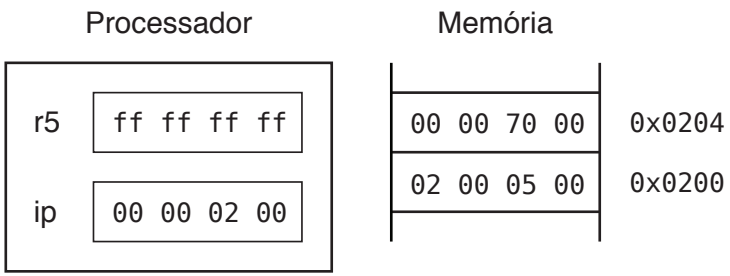
A sequência de passos realizados pelo processador a partir dessa configuração será:

1. *Busca de instrução*: o processador faz um acesso à memória para ler a palavra de memória apontada por `ip` (endereço `0x200`) e coloca o valor lido (`0x02000500`) no registrador interno `ir`. O registrador `ip` é avançado do número de bytes lidos: `ip` passa a valer `0x204`. O processador decodifica a instrução (examina o código `0x02`) e determina que é uma instrução do tipo "carrega registrador com valor constante" e possui duas palavras. Processador faz novo acesso à memória para ler a segunda palavra da instrução, na posição de memória apontada por `ip` (endereço `0x204`) e coloca o valor lido (`0x7000`) no registrador `r5`. O registrador `ip` é avançado do número de bytes lidos: `ip` passa a valer `0x208`.

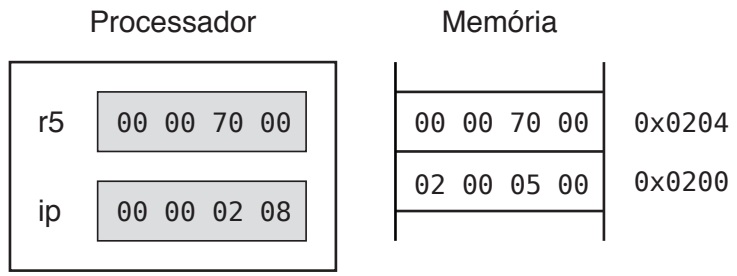
Exemplo

2. *Busca de operando*: nada a fazer neste passo no caso desta instrução.
3. *Execução*: nada a fazer neste passo no caso desta instrução.
4. *Armazenamento de resultado*: nada a fazer neste passo no caso desta instrução.

Exemplo



Estado antes da execução



Estado após a execução

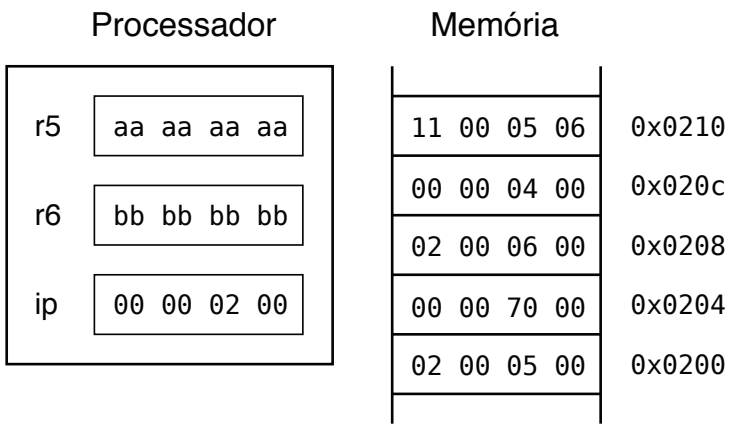
Um pequeno programa

Vamos escrever um "programa" simples, para calcular a soma dos valores `0x7000` e `0x400` e armazenar o resultado no registrador `r5`.

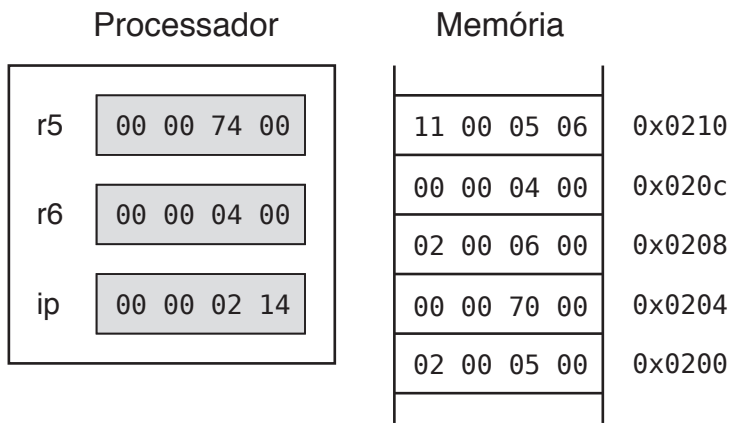
Um pequeno programa

```
0x02000500  carrega r5 com valor contido
0x00007000  nesta segunda palavra da instrução
0x02000600  carrega r6 com valor contido
0x00000400  nesta segunda palavra da instrução
0x11000506  soma r5 com r6 e coloca o resultado em r5
```

Um pequeno programa



Estado antes da execução



Estado após a execução