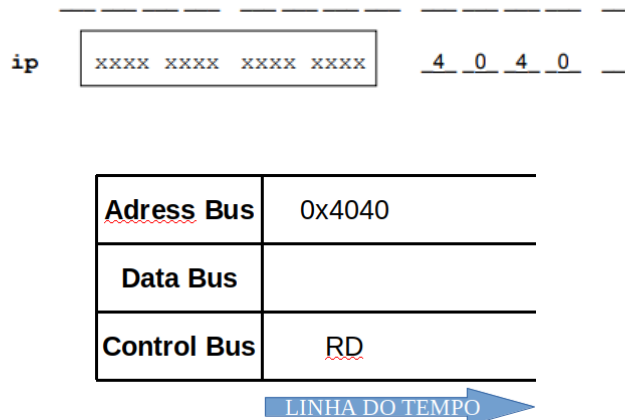


## SC - PASSOS DE INTERPRETAÇÃO DE UM CICLO DE INSTRUÇÃO

**1º** Lê-se, read-RD, o valor do registo %ip, Instruction Pointer – IP, e vê-se para que endereço de memória é que está a apontar. Como se procedeu a uma leitura o **Control Bus** tem o sinal **RD**;

Neste caso o endereço de memória para onde está a apontar o registo %ip é **0x4040** e esse mesmo valor é escrito simultaneamente no **Adress Bus**.

### Banco de registos

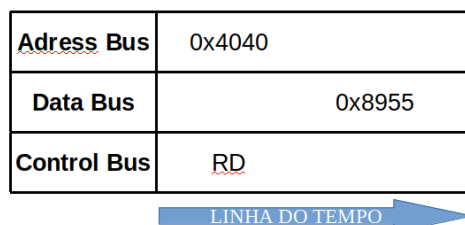


**Nota:** A leitura do Instruction Pointer é sempre o 1º passo no ciclo de execução de uma máquina, sem esta leitura a máquina não sabe que instrução executar! Como tal torna-se obrigatório, antes de tudo, a leitura do registo %ip no banco de registos.

**2º** No endereço de memória 0x4040 o valor existente é 55, mas como se trata de uma máquina de 16bits a capacidade do Data Bus é de 2 Bytes, logo também é capturado o valor da posição de memória imediatamente a seguir. Atenção à ordem da captura destes valores, como se trata de uma máquina little endian(1) a ordem de captura é 1º o valor 89 e a seguir o valor 55. Desta forma o valor obtido na memória para o Data Bus é **0x8955**.

### Memória

0x4040	0101 0101	5 5	—
1	1000 1001	8 9	—
0x4042	1110 0101	e 5	—



Obs: Definição de Adress Bus, Data Bus e Control Bus(2).

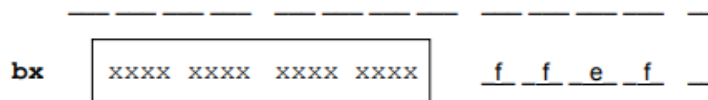
**3º** Descodificada a instrução, que já nos é dada no enunciado, temos então:

```
addw %bx, -8(%bp)
```

**4º** A instrução dada é **addw**, uma adição. Para esta mesma instrução de adição, são necessários “pelo menos” 2 operandos.

**%bx** é um registo, que tem guardado nele o valor **0xFFFF**, registo esse que é o 1º operando. O 2º operando da instrução encontra-se no endereço de memória (**-8(%bp)**). Este endereço de memória, (**-8(%bp)**), nada mais é, que uma soma de 2 valores, i é, valor que está num registo + o Displacement 8 bits, cujo seu resultado será o endereço de memória onde está o 2º operando da instrução **addw**.

## Banco de registros

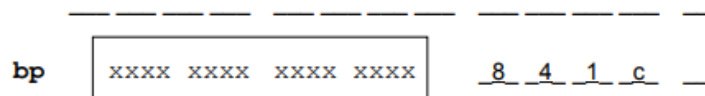


**4.1°** Calcular o endereço de memória, **(-8(%bp))**, onde se encontra o 2º operando da instrução **addw**:

a)

**%bp** obtém-se indo ao banco de registos e capturando o seu valor que neste caso é o valor em hexadecimal **0x841C**;

## Banco de registros



b)

Agora tem-se que:

```

(-8(%bp)) <=> %bp + (-8) <=> %bp - 8 <=> %bp <=> 0x841C
      -8      -8
      ...      0x8414

```


c)

O valor do 2º operando da instrução **addw** encontra-se no endereço de memória **0x8414**.

**5º** Neste momento sabe-se em que endereço da memória é que se encontra o valor do 2º operando da instrução **addw, 0x8414**, então da-se a ordem de leitura do conteúdo existente nesse endereço de

memória, enviando o sinal de leitura, *read* – **RD**, pelo **Control Buss** e passado o respetivo endereço de memória, a ser lido, pelo **Adress Bus**;

<b>Adress Bus</b>	0x4040	0x8414
<b>Data Bus</b>	0x8955	
<b>Control Bus</b>	<u>RD</u>	RD

LINHA DO TEMPO 


**6º** É efetuada a ordem de leitura passada pelo **Control Bus**, **RD**, do endereço passado pelo **Adress Bus**, **0x8414** (não esquecer a política *little endien*).

## Memória

<b>0x8414</b>	0001 0010	<u>1</u> <u>2</u> —
<b>5</b>	0100 0000	<u>4</u> <u>0</u> —
<b>0x8416</b>	0001 0100	<u>1</u> <u>4</u> —

∴ É neste momento que se obtém então, de forma explícita, o valor do 2º operando da instrução que foi dada, **addw**. Valor esse que é **0x4012** e que é imediatamente passado pelo **Data Bus**.

<b>Adress Bus</b>	0x4040	0x8414
<b>Data Bus</b>	0x8955	0x4012
<b>Control Bus</b>	<u>RD</u>	RD

LINHA DO TEMPO 

**7º** Neste estado do ciclo de todo o processamento, já se sabe qual é a instrução a executar e os dois operandos envolvidos, de forma explícita, desta mesma instrução.

**addw %bx, -8(%bp)**

...que é o mesmo que ler...

**addw (0xFFFF) (0x4012)**

...que equivale a ...


$$\begin{array}{r} 0xFFFF + 0x4012 \Rightarrow 0xFFFF \\ +0x4012 \\ \hline 0x14001^* = 0x4001 \end{array}$$

\*não dá OVERFLOW porque é a soma de um negativo (0xFFFF) com um positivo. Ignora-se o último bit de transporte porque trata-se de uma máquina de 16bits.

**8º** Obtido o resultado da instrução aplicada aos operandos anteriormente determinados resta apenas escrever o resultado obtido, **0x4001**, no endereço de memória anteriormente calculado do 2º operando(3), **0x8414**.

Para que tal aconteça o valor a ser escrito/guardado, **0x4001**, é passado pelo **Data Bus**, o local na memória onde será escrito esse resultado, **0x8414**, é passado pelo **Address Bus**, e a ordem de escrita do resultado é passada pelo **Control Bus** pelo envio do sinal de escrita, **read-RD**.

<b>Address Bus</b>	0x4040	0x8414	0x8414
<b>Data Bus</b>	0x8955	0x4012	0x4001
<b>Control Bus</b>	RD	RD	<u>WR</u>


 LINHA DO TEMPO

**9º** Incrementa o Instruction Pointer (registo %ip) para que se proceda à execução da próxima instrução e se dê início novamente a todo este ciclo (do passo 1 ao 9).

-----FIM DA EXECUÇÃO DO CICLO\*\*-----

**REGISTOS MODIFICADOS: %IP**  
**CÉLULAS DE MEMÓRIA MODIFICADAS(4): 0x8414 e 0x8415**

(1) – Em little endien o byte menos significativo do barramento refere-se ao conteúdo da célula de memória com o endereço mais baixo; No caso do big endien é o contrario;

(2) –

**Adress Bus** – Barramento de endereços é responsável por transportar, neste caso, 16bits de cada vez e apenas durante o período de tempo em que esses valores estiverem ativos no barramento;

**Data Bus** – Barramento de Dados é responsável por transportar, neste caso, 16 bits de cada vez, e apenas durante o período de tempo em que esses valores estiverem no barramento;

**Control Bus** – Barramento de controlo é responsável por transportar os sinais de controlo que forem necessários, nomeadamente, escrita ou leitura(Read -RD ou Write – WR).

(3) – O local onde é guardado o resultado obtido é indicado no enunciado, logo no capítulo 1;

(4) – São modificadas 2 células de memória porque como indicado acima, trata-se de uma maquina de 16bits o que implica alocação constante de 2 bytes de memória seguidos, alem de que o resultado obtido na instrução dada ocupa esses 2 Bytes e a ordem de inserção na memória é determinada pela forma de como é transportada a informação, que neste caso é sob a forma *little endien*:

#### RESULTADO OBTIDO DA INSTRUÇÃO: 0x4001

1º byte(byte menos significativo do barramento):0x\*\*01  
2º byte: 0x40\*\*

Estado da memória **antes** de guardar o resultado:

Memória		
0x8414	0001 0010	<u>1</u> <u>2</u>
5	0100 0000	<u>4</u> <u>0</u>
0x8416	0001 0100	<u>1</u> <u>4</u>

Estado da memória **depois** de guardar o resultado:

Memória		
0x8414	0001 0010	<u>0</u> <u>1</u>
5	0100 0000	<u>4</u> <u>0</u>
0x8416	0001 0100	<u>1</u> <u>4</u>