

# Exercício Prático 5 – Arq. de Computadores 2

## Romanelli

**Objetivo:** Neste relatório iremos reforçar os conceitos MIPS, MFLOPS, frequência, período e ciclos por instrução (ou CPI).

### Introdução:

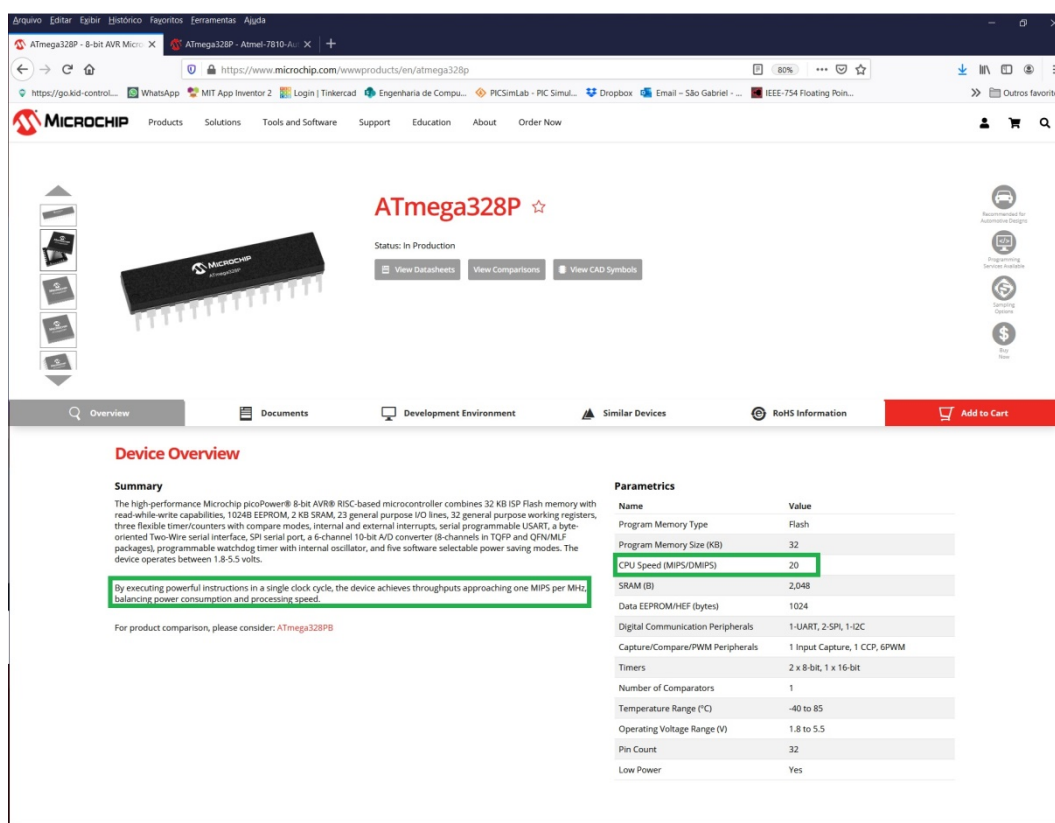
Avaliar um programa medindo o tempo através de um código em alto nível não é uma boa estratégia, incluído neste tempo em geral também estão os tempos necessários do SO, de I/O e também do compilador utilizado.

A melhor estratégia seria construir um programa em uma linguagem de baixo nível (ou a linguagem mais próxima possível do Hardware do processador) e assim, conhecendo-se a quantidade das instruções deste programa, a quantidade de ciclos necessários à execução de cada uma das diferentes instruções deste programa e a frequência de operação do processador, determinaríamos o CPUtime ou o tempo real de execução do código, desconsiderando todas as variáveis incorporadas ao tempo de execução e já mencionadas. Estas são as variáveis que encontramos na Equação de Performance da CPU e que iremos utilizar daqui para frente.

Contudo, nesta experiência não adotaremos esta estratégia acima, nós a iremos realizar durante as aulas teóricas e quando estivermos escrevendo programas em linguagem de baixo nível, mas por agora, iremos construir um método de avaliação e aprender a lidar com os conceitos de MIPS, MFLOPS, quantidade de instruções, ciclos por instrução e frequência.

### Experiência 1 – Avaliação do Arduino

Se olharmos o datasheet do fabricante do ATM328P, que é o processador da placa do Arduino, podemos ver a velocidade do processador divulgada e que é da ordem de 1 MIPS/MHz.



The screenshot shows the Microchip website for the ATmega328P microcontroller. The page is titled "ATmega328P" and includes a summary of the device's features. A green box highlights a key performance metric: "By executing powerful instructions in a single clock cycle, the device achieves throughput approaching one MIPS per MHz, balancing power consumption and processing speed." Below this, a table of parameters is provided.

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	20
SRAM (B)	2,048
Data EEPROM/HEF (bytes)	1024
Digital Communication Peripherals	1-UART, 2-SPI, 1-I2C
Capture/Compare/PWM Peripherals	1 Input Capture, 1 CCP, 6PWM
Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	32
Low Power	Yes

Como o cristal existente na placa do Arduino é de 16MHz, podemos assumir então uma velocidade de 16 MIPS.

$$1 \text{ MIPS/MHz} * 16 \text{ MHz} = 16 \text{ MIPS}$$

Vamos submeter o arduino ao seguinte código:

EP05\_teste \$

```
long c;
byte i, j;
long inicio, fim, tempo;
void setup() {
  Serial.begin(9600);
}

void loop() {
  i=1;
  j=3;
  inicio=micros();
  for(c=0;c<1000000;c=c+1) i=j;
  fim=micros();
  tempo=(fim-inicio);
  Serial.print("tempo= ");
  Serial.println(tempo);
}
```

A função micros() retorna o tempo em microssegundos desde que o arduino foi iniciado com um erro, para o arduino UNO, de aproximadamente 4 us.

Assim, se considerarmos que iremos realizar um laço de 1000000 vezes (ou 1M vezes) a variável tempo conterá o tempo que este laço irá demorar. Como está dentro da função loop() iremos ver repetidas vezes este valor. Iremos tomar sempre uma média de uns 10 valores em todas as medidas que iremos realizar.

Vamos realizar os testes em apenas instruções lógicas e aritméticas, mas iremos avaliar os tempos para diferentes tipos (byte, int, float). Assim, também iremos conseguir avaliar, no caso de estarmos com operandos do tipo float, o MFLOAT.

Para criarmos um Baseline, ou uma referencia, deveremos incluir no laço uma atribuição, para evitarmos os erros proporcionados por ela. Resumindo, em todos os tempos das operações medidas iremos subtrair os tempos gastos pelo for e pela atribuição, ficando apenas com os tempos da

operação. Lembre-se que estamos com um código em alto nível e existe no meio do processo um compilador que irá traduzir o nosso código para a linguagem de baixo nível do ATM328P.

Para criarmos este Baseline então, usaremos a seguinte linha de código:

*for(c=0;c<1000000;c=c+1) i = j ;*

Agora você deve preencher o seguinte quadro:

Tipo	Tempo base	Use para o teste ( i = i op 3 )			Use para o teste ( i = i op j )		
		Soma	Or	Mult	Soma	Or	Mult
byte							
int							
float			XXXX			XXXX	

Uma vez de posse dos valores, calcule a quantidade de MIPS de cada instrução para cada tipo de dado. Obviamente para os valores em float estaremos calculando o MFLOAT.

Como estaremos executando 1M instruções (1000000 instruções) a conta irá ficar direta.

Vamos considerar as seguintes leituras (são valores reais medidos em um arduino):

Tipo	Tempo base	Soma	Mult.	Or	Soma	Mult.	Or
byte	754700	1006220	1069104	817580	1006230	1006220	817580

Cálculos:

Tempo de 1000000 somas = 1006220us – 754700us = 251520 us, mas 251520us = 0,251520 s

Assim, realizamos 1000000 somas em 0,251520 segundos ou realizamos 1M somas em 0,251520.

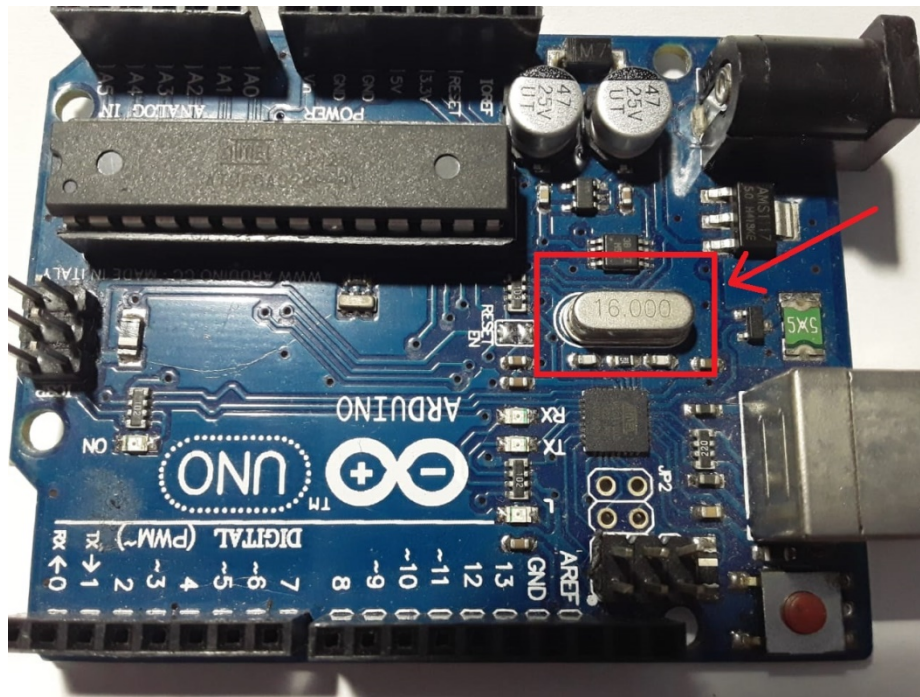
Podemos dizer que estamos com  $1 / 0,251520 \text{ MIPS} = 3,9758 \text{ MIPS}$ .

Com os cálculos realizados preencha a seguinte tabela comparativa:

Tipo	MIPS ( ATM328P )					
	Constante ( Ex.: i=i op 3 ; )			Variável ( Ex.: i=i op j ; )		
	Soma	Or	Mult	Soma	Or	Mult
byte	3,9758					
int						
Tipo	MFLOPS ( ATM328P )					
	Constante			Variável		
	Soma	Or	Mult	Soma	Or	Mult
float		XXXXXXX			XXXXXXX	

Podemos ainda tirar algumas conclusões mais interessantes:

O processador ATM328P quando na placa do Arduino está submetido a um cristal que irá fornecer o sinal de clock e que está oscilando a 16 MHz. Veja como identificar o cristal na placa do arduino:



Conhecendo-se então a frequência de clock do processador podemos fazer as seguintes contas:

Frequência de 16MHz =  $16 \times 10^6$  ciclos/segundos

Mas para o caso do tipo byte e executando uma soma medimos o seguinte:

O processador irá realizar  $1 \times 10^6$  somas em 0,251520 segundos

**Pergunta: Quantos ciclos serão necessários para que ele realize uma soma?**

$$\text{Quantidade de ciclos para cada soma} = \frac{(\text{tempo para realizar uma soma})}{(\text{tempo de um ciclo})}$$

Ou

$$\text{Quantidade de ciclos por soma} = (\text{tempo para realizar uma soma}) * \frac{1}{(\text{tempo de um ciclo})}$$

Mas (  $1 / \text{tempo para realizar um ciclo}$  ) é o mesmo que (1/período da onda), ou seja é a frequência da onda.

Com isso podemos reescrever como:

Quantidade de ciclos = tempo para realizar uma soma \* frequência da onda

$$\text{Quantidade de ciclos por soma} = \frac{0.251520 \text{ segundos}}{1 \times 10^6 \text{ somas}} * 16 \times 10^6 \text{ ciclos / segundos}$$

$$\text{Quantidade de ciclos por soma} = 0,251520 * 16$$

$$\text{Quantidade de ciclos por soma} = 4,024 \text{ ciclos por soma}$$

O que acabamos de calcular chama-se CPI e quer dizer ciclos por instrução, no caso específico a instrução é uma soma.

Assim, de posse deste número você acabou de formular a equação de performance da CPU:

$$\text{CPUtime} = \text{IC} * \text{CPI} * 1/f$$

Onde IC = Quantidade de instruções

CPI = Número de ciclos por cada instrução

1/f = período da onda ou simplesmente 1 / frequência do clock

Vamos verificar se a equação funciona:

IC = quantidade de instruções =  $1 \times 10^6$ , neste caso estamos tratando das somas realizadas

CPI = 4,024 ciclos por instrução, neste caso a quantidade de ciclos por cada soma que calculamos

1/f =  $1 / 16 \times 10^6$  Hz, ou a frequência que o processador está submetido

Se efetuarmos a conta teremos:

$$\text{CPUtime} = 1 \times 10^6 * 4,024 * 1 / 16 \times 10^6 = 0,251520$$

Observe que foi o tempo medido de execução das  $1 \times 10^6$  instruções ou somas.

Você agora deverá preencher a tabela abaixo com os CPI's de cada uma das instruções executadas no Arduino:

Tipo	CPI					
	<i>Soma</i>	<i>Or</i>	<i>Mult</i>	<i>Soma</i>	<i>Or</i>	<i>Mult</i>
byte	<b>4,024</b>					
Int						
float		XXXX			XXXX	

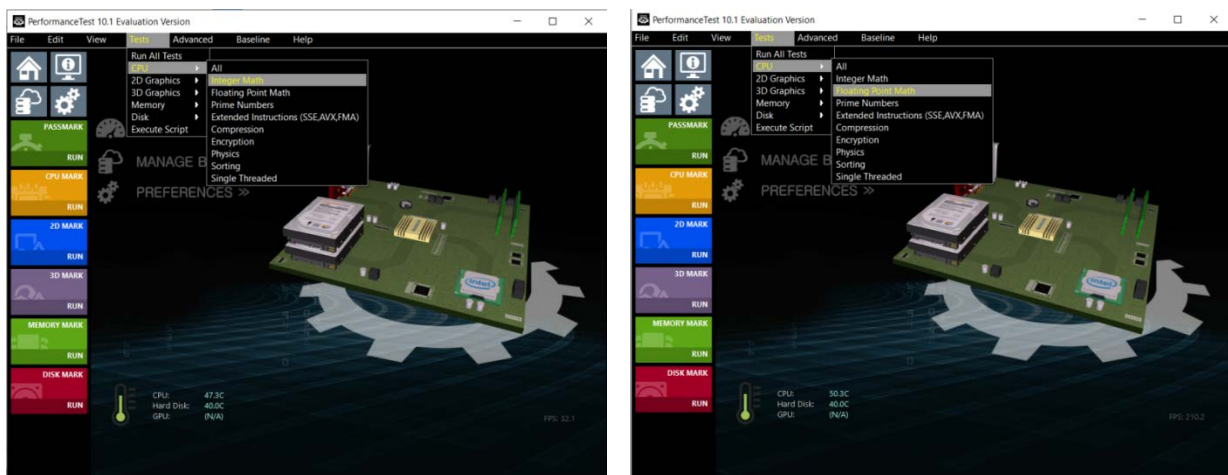
## Experiência 2 – Seu PC

Agora faremos a mesma coisa, porém com a sua máquina. Você verá como o SO e o compilador podem mascarar os valores dos MIPS, MFLOPs e o CPI das instruções. Para realmente avaliarmos estes valores e elaborar a equação de performance da CPU deveríamos fazer todas as avaliações sobre a linguagem de mais baixo nível possível sem interferências destes elementos. Esta atividade será melhor realizada em grupo ou, se possível, executada em duas máquinas diferentes.

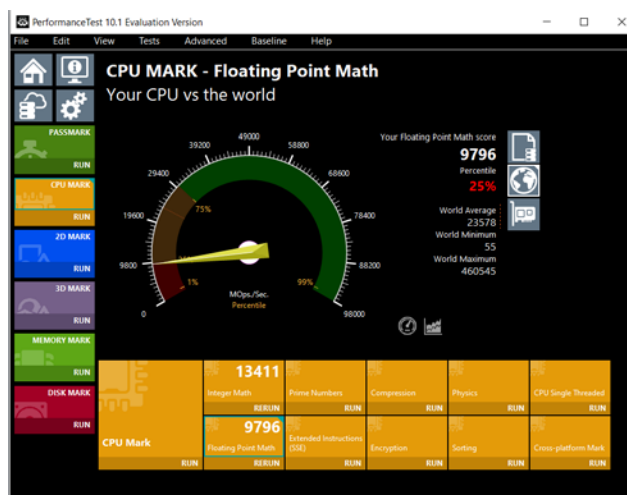
- 1) Vamos usar um programa de benchmark simples para termos um valor mais próximo da realidade dos MIPS e MFLOPs da sua máquina. Diversos programas de avaliação de desempenho oferecem este número, entretanto separei um programa mais simples que irá mostrar estes números de uma forma mais clara. Vamos usar o:

<https://www.passmark.com/products/performance-test/download.php>

- 2) Rode o programa avaliando especialmente a CPU em particular os testes em números inteiros e em ponto flutuante:



- 3) Você irá obter a tela a seguir com os valores avaliados para a sua CPU, quer seja para operações com números inteiros e operações com números em ponto flutuante. A unidade indicada é em Mops/s ou seja, Milhões de operações por segundo (MIPS).



- 4) Agora vamos realizar alguns testes procedendo conforme fizemos no caso do Arduino, mas lembre-se que estaremos fortemente amarrados ao SO, ao compilador que estamos utilizando e eventualmente às entradas e saídas no momento do teste. Abra o compilador C e execute o seguinte programa:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main()
6  {
7      clock_t inicio, fim, T;
8      float Tempo, media=0;
9      int c;
10     int k, i=5, j=3;
11     T=CLOCKS_PER_SEC;
12     for (k=1;k<=10;k=k+1) // apenas para realizar a medida 10 vezes
13     {
14         inicio=clock(); // Marca o tempo inicial
15         for (c=1;c<=1000000;c=c+1)i=j; // executa o loop
16         fim = clock(); //Marca o tempo final
17         Tempo =( (fim - inicio)*1000/CLOCKS_PER_SEC); //Calcula tempo final - tempo inicial
18         printf("\nTempo : %g ms.", Tempo);
19         media=media+Tempo;
20     }
21     printf("\nTempo gasto media: %g ms.", media/10);
22 }

```

Se possível, procure ajustar o compilador para compilar para i386, para tornarmos a avaliação mais comum a todos os participantes do grupo. Coloque a variável de controle do laço do tipo register, isso faz com que tenhamos um registrador fixo para ela e podemos ganhar algum tempo (troque por int e faça a medida para comprovar esta afirmação!).

Faça agora os mesmos procedimentos realizados para o Arduino e preencha as tabelas a seguir. Assim como no Arduino, inseri alguns valores reais que obtive

Tipo	Tempo base	Use para o teste ( i = i op 3 )			Use para o teste ( i = i op j )		
		Soma	Or	Mult	Soma	Or	Mult
char	4,9 ms	25 ms					
Int							
float			XXXXXX			XXXXXX	

Tempo para 10.000.000 somas = 25ms – 4,9ms = 21,1 ms

Cálculo de MIPS =>  $10 \cdot 10^6 / 21,1 \cdot 10^{-3}$   
 $= 10 / 21,1 \cdot 10^{-3}$  MIPS  
 $= 473,93$  MIPS



Tipo	MIPS ( Seu PC )					
	Constante			Variável		
	Soma	Or	Mult	Soma	Or	Mult
char	473,93					
int						

Tipo	MFLOPS ( Seu PC )					
	Constante			Variável		
	Soma	Or	Mult	Soma	Or	Mult
float		XXXXXX			XXXXXX	

Finalmente, calcule o CPI para cada uma das operações. Para o cálculo do CPI será necessário a frequência da sua máquina. Você pode obter este valor através do próprio programa de benchmark ou olhando no seu próprio sistema (tecla “Windows+x” e escolha “sistema”).

The screenshot shows the PerformanceTest 10.1 Evaluation Version interface. The 'SYSTEM INFORMATION' tab is selected, indicated by a red arrow. The 'This Computer' section is highlighted with an orange box, and the 'Measured Speed' value of 2993.3 MHz [Turbo: 3192.8 MHz] is pointed to by an orange arrow. The interface also shows benchmark results for PASSMARK, CPU MARK, 2D MARK, 3D MARK, MEMORY MARK, and DISK MARK, each with a 'RUN' button.

No caso acima a frequência seria de 2993,3 MHz

$$\text{CPI} = 21,1 \times 10^{-3} / 10 \times 10^6 * 2993,3 \times 10^6$$

$$\text{CPI} = 6,315863$$

Tipo	CPI					
	Soma	Or	Mult	Soma	Or	Mult
byte	6,315863					
Int						
float		XXXXX			XXXXXX	

Uma vez que você executou o teste de performance utilizando o programa de teste baixado, vamos comparar a sua máquina e dos outros componentes do grupo, preencha a tabela a seguir de acordo



com o número de membros do grupo. Você deverá considerar a pior máquina como o padrão e calcular o speedup de todas as outras máquinas sobre a pior do grupo.

Para o cálculo considere os valores obtidos nas medidas das operações inteiras e de ponto flutuante. Considere por exemplo que iremos submeter as máquinas a 1.000.000 instruções inteiras e de ponto flutuante. Desta forma, se multiplicarmos a velocidade lida por 1.000.000 instruções teremos o tempo de execução desta quantidade de instruções e assim podemos calcular o Speedup, que é uma relação de tempo.

Identificação da máquina (processador, frequência de clock, SO e Compilador usado)	Prog. em C		Performance Test	
	Speed up (inteiros)	Speed up (FP)	Speed up (inteiros)	Speed up (FP)
<i>Escreva aqui a máquina padrão ou a pior máquina testada.</i>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>

#### Testes por SO, Compilador e processador

Identificação do processador, frequência de clock, compilador	Prog. em C (inteiros)		Speed up
	<i>Escreva aqui o <b>SO</b> utilizado</i>	<i>Escreva aqui o <b>SO</b> utilizado</i>	

Identificação do processador, frequência de clock, SO	Prog. em C (inteiros)		Speed up
	<i>Escreva aqui o <b>Compilador</b> utilizado</i>	<i>Escreva aqui o <b>Compilador</b> utilizado</i>	

Identificação do SO e Compilador	Prog. em C (inteiros)		Speed up
	<i>Detalhes da <b>Máquina</b></i>	<i>Detalhes da <b>Máquina</b></i>	

## **O que apresentar para este relatório:**

- Todas as tabelas preenchidas com as medidas realizadas
- um printscreen de:
  - pelo menos um teste no arduino;
  - pelo menos dois diferentes testes do programa em C em máquinas diferentes;
  - pelo menos dois testes com o programa de benchmarks baixado em máquinas diferentes.