Arquitetura de Computadores
Professor: Leandro Coelho
Aluna: Ana Paula Santos Pinheiro

Part A: (Refer to the appendix for help with CPU instructions)

1. Crie uma instrução que mova o número 5 para o registrador R00.

   MOV #5, R00

2. Crie uma instrução que mova o número 7 para o registrador R01.

   MOV #7, R01

3. Crie uma instrução que adicione o conteúdo de R00 e R01 e Anote em qual registrador o resultado foi colocado.

   ADD R00, R01

4. Crie uma instrução que empurre o valor do registrador acima para o topo da pilha de programa e execute-a.

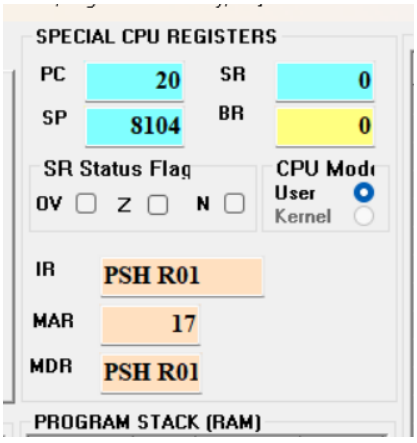
   PSH R01

5. Crie uma instrução para empurrar o número 2 para o topo da pilha de programa e execute-a.

   PSH #2

6. Crie uma instrução para saltar incondicionalmente para a primeira instrução.
   a. Observe o valor no registrador PC. Este é o endereço da próxima instrução a ser executada. Faça uma anotação sobre qual instrução ele está apontando

   JEQ 00



7. Crie uma instrução para retirar o valor do topo da Pilha de Programa para o registrador R02.

   POP R02

8. Crie uma instrução para retirar o valor do topo da Pilha de Programa para o registrador R03.

   POP R03

9. Execute a última instrução novamente. O que aconteceu? Explique.
   a. R= O programa primeiro armazenou o resultado 0 de 5 no registrador R00. Em seguida, somou 7 + 5, o que deu 12, e armazenou o resultado no registrador R01. Depois, ele colocou os dois valores no topo da pilha. Em seguida, os valores no topo da pilha foram armazenados nos registradores R2 e R3. Inicialmente, não consegui armazenar os valores da pilha logo, pois o jump estava atrapalhando. Então, coloquei o jump na última linha, e aí consegui fazer sem problemas.

      Ao colocar o jump na última linha, o fluxo do programa foi controlado corretamente, permitindo que os valores fossem armazenados nos registradores R2 e R3 após o processo de manipulação da pilha, sem causar interferência no processo de execução.

10. Crie uma instrução de comparação(Insira manualmente dois valores iguais), que compara os valores nos registradores R04 e R05.
    a. Quais das bandeiras de status OV/Z/N estão ativadas (ou seja, a caixa está marcada)?

       R= Z

11. Insira manualmente um valor no primeiro registrador maior do que o do segundo registrador, e execute novamente a instrução de comparação.
    a. Quais das bandeiras de status OV/Z/N estão ativadas?

       R= N

12. Insira manualmente um valor no primeiro registrador menor do que o do segundo registrador, e execute a instrução de comparação mais uma vez.
    a. Quais das bandeiras de status OV/Z/N estão ativadas?

       R= OV

13. Crie uma instrução que vá para a primeira instrução se os valores nos registradores R04 e R05 forem iguais.

    0030 MOV #12, R04

    0036 MOV #10, R05

    0042 CMP R04, R05

14. Teste a instrução acima colocando manualmente valores iguais nos registradores R04 e R05, em seguida, execute primeiro a instrução de comparação e depois execute a instrução de salto.
    a. Se funcionar, a primeira instrução deve ser destacada.

	LAdd	Instruction	Base	T
00	0000	MOV #5, R00	0000	0
06	0006	MOV #7, R01	0000	0
12	0012	ADD R00, R01	0000	1
17	0017	PSH R01	0000	0
20	0020	PSH #2	0000	0
24	0024	POP R02	0000	0
27	0027	POP R03	0000	0
30	0030	MOV #10, R04	0000	0
36	0036	MOV #10, R05	0000	0
42	0042	CMP R04, R05	0000	3
47	0047	JEQ 0	0000	2

15- Agora que você tem algum entendimento sobre as instruções básicas da CPU e é capaz de programar o simulador, aqui vai um pequeno desafio para você: preparar um pequeno loop de programa. Os loops de programa são extremamente úteis e são usados com muita frequência pelos programas de computador. Veja o que você precisa fazer:

1. Crie uma instrução que mova o número 0 para o registrador R01.

MOV #0, R01

2. Crie uma instrução que adicione o número 1 ao registrador R01.

   Program teste:

   INC R01

3. Crie uma instrução que compare o número 10 com o registrador R01.

   CMP #10, R01

4. Crie uma instrução que pule de volta para a instrução 2 acima se R01 não for igual ao número 10.

   JNE 6

5. Crie uma instrução HLT.

   HLT

Anote as instruções 1 a 5 que você criou acima na caixa abaixo:

6. Começando pela instrução 1, execute manualmente as instruções 1 a 4 uma após a outra. O que aconteceu quando você executou a instrução 4?

   R= O que acontece na instrução 4 é que o jump vai voltar para a segunda instrução, criando um loop infinito, porque a condição nunca será atendida. O valor de R01 será sempre 1, já que na instrução 2 o número 1 é adicionado a R01. Como a comparação é feita com o número 10, e R01 sempre terá o valor 1, a condição "se R01 não for igual a 10" sempre será verdadeira, fazendo com que o código continue repetindo o loop e nunca chegue à instrução HLT, ou seja, o programa nunca vai terminar.

Parte B: (Consulte o apêndice para ajuda com as instruções da CPU)

1. Produza o código para uma instrução condicional tal que, se o valor no registrador R02 for maior que o valor no registrador R01, então o registrador R03 será definido como 8. Teste no simulador.

   0000 MOV #2, R01

   0006 MOV #3, R02

   0012 CMP R01, R02

   0017 JGT 31

   0021 MOV #7, R03

   0027 JMP 0

   0031 MOV #8, R03

   R= No código, eu fiz uma modificação para poder verificar as condições. Por exemplo, se R2 for maior que R1, ele vai dar um jump para a linha 31, ou seja, vai promover o valor de R3. Porém, agora, se R2 não for maior que R1, ele vai continuar e moverá o valor 7 para o R3, para que possamos saber essa condição. Depois, ele vai pular para o início, o que faz com que o programa entre em um loop infinito.

2. Produza o código para um loop que se repete 5 vezes, onde o valor do registrador R02 é incrementado em 2 a cada repetição do loop. Teste no simulador.

   R= No código abaixo, eu usei o registrador R0 como o valor limite, que é 5, para contar o número de vezes que o loop será executado. Assim, o loop será executado 5 vezes, e cada vez que o loop acontece, o valor de R2 é incrementado de 2 em 2. O registrador R1 também sofre um incremento a cada iteração do loop, adicionando 1. A comparação é feita entre R0 e R1: se R1 for menor que R0, o código dará um jump de volta para o início do loop e continuará a execução. Quando R1 e R0 se tornarem iguais, o jump não será executado, e o programa terminará.

3. Os números 4, -3, 5 e -6 são empilhados manualmente na pilha, nessa ordem. Produza o código para uma rotina que retire dois números do topo da pilha, multiplica-os e empilha o resultado de volta no topo da pilha. A rotina repete isso até que reste apenas um número no topo da pilha. Teste o código no simulado:

0000 MOV #4, R00

0006 PSH R00

0009 MOV #-3, R00

0015 PSH R00

0018 MOV #5, R00

0024 PSH R00

0027 MOV #-6, R00

0033 PSH R00

0036 MOV #0, R00

0042 MOV #6, R01

0048 POP R02

0051 POP R03

0054 MUL R02, R03

0059 PSH R03

0062 INC R04

0065 INC R04

0068 CMP R04, R01

0073 JGT 48

R= No meu código, eu comecei a empilhar manualmente os números na pilha. Depois, retirei os dois primeiros valores da pilha e os armazenei nos registradores **R2** e **R3**. Realizei a multiplicação desses valores e armazenei o resultado em **R3**. Usei o registrador **R4** como contador, que foi incrementado até o limite de **6**, representando o número total de multiplicações. A cada iteração, retirei dois números da pilha, os multipliquei e empilhei o resultado novamente. Esse processo continuou até restar apenas um número na pilha, que é o resultado final da multiplicação de todos os valores.