m6_060924.txt
este arquivo: m6_060924.txt
==========
TEMAS e MATERIAIS PARA ESTUDO - 060924
Consulte também: https://pater.web.cip.com.br/MaquInfo/

Em particular, para esta e para as aulas subsequentes:
https://pater.web.cip.com.br/MaquInfo/ch09.html ("PROCESSANDO PADRÕES SIMBÓLICOS")
https://pater.web.cip.com.br/MaquInfo/ch10.html#idp1010001 ("MULTIPLICANDO DOIS INTEIROS")

Esquema: https://pater.web.cip.com.br/SI2024/cdpam2024.pdf

Acompanhe as leituras
"Processando Padrões Simbólicos" https://pater.web.cip.com.br/MaquInfo/ch09.html
"Multiplicando dois inteiros" https://pater.web.cip.com.br/MaquInfo/ch10.html#idp1010001

com o esquema CDPAM https://pater.web.cip.com.br/SI2024/cdpam2024.pdf
(Computador Digital de Programa Armazenado em Memória)

====
Recapitulando e ampliando parte da aula passada (040924), imagine um jogo com quatro caixas, chamadas A, B e C e Z, inicialmente vazias, ou com um número desconhecido de objetos (bolas, pedras, moedas, palitos).

Variando a quantidade de objetos em cada caixa (observadas simultaneamente) podemos modificar e registrar o "estado" (situação) geral do jogo em determinado momento.

O jogo progride através da repetição de uma sequência de movimentos idênticos, cada um deles compreendendo a realização de um ato predeterminado capaz de alterar o estado do jogo a cada momento.

Cada ato (ou "jogada") corresponde a uma ação elementar, do pequeno conjunto de ações disponíveis. O passo-a-passo do jogo encadeia as jogadas obtendo efeitos sobre o estado do jogo a cada momento.

Os passos modificam o estado do jogo, mas não podem modificar a alternância essencial de movimentos, sempre a mesma do início ao fim do jogo.

Os atos distintos (com efeitos distintos sobre o estado do jogo) são dinamizados através daqueles movimentos repetidos, através de sequências diferentes de atos.

Dito de outro modo, o encadeamento dos efeitos sobre o estado do jogo depende de uma "sequência predeterminada de instruções", de um "programa".

Nos jogos de tabuleiro, por exemplo, estamos acostumados com a alternância entre jogadores: eu jogo, você joga, eu jogo, você joga, etc.. Um jogo é diferente do outro, mas todos seguem esta mesma regra.

Também sabemos como produzir textos muito diferentes com as mesmas regras gerais: escrevemos da esquerda para a direita, de cima para baixo, com o verso da página do lado esquerdo do caderno.

Há "jogos de escrita" com outras regras, certamente, mas o argumento é o mesmo.

As caixas A, B, C e Z, como dissemos, estão "vazias" ao início do jogo, ou, o que é indiferente para o nosso jogo, com uma quantidade desconhecida de objetos dentro delas.

A caixa vazia aqui representa uma caixa com o valor zero dentro dela.

Nota: Não é assim que representamos hoje o zero na escrita de números em geral. O zero é normalmente representado por alguma marca ou símbolo, e não por uma ausência.

==========

MOVIMENTOS E ESTADOS SUCESSIVOS DO JOGO DAS CAIXAS
(JNZ nn significa [JUMP if NOT ZERO to nn], IR PARA O ENDEREÇO nn SE O ÚLTIMO RESULTADO NÃO FOI ZERO)

linha	caixa A	caixa B	caixa C	caixa Z	
00 A = 0	0	?	?	?	
01 B = 7	0	7	?	?	
02 C = 3	0	7	3	?	
03 Z = 0	0	7	3	0	
					fim da inicialização
04 A = A + B	7	7	3	0	
05 C = C - 1	7	7	2	0	
06 JNZ 04	7	7	2	0	
04 A = A + B	14	7	2	0	
05 C = C - 1	14	7	1	0	

06 JNZ 04	14	7	1	0
04 A = A + B	21	7	1	0
05 C = C - 1	21	7	0	1
06 JNZ 04	21	7	0	1
07 RET (sair)	21	7	0	1

Resultado na caixa A: 21
Produto do valor na caixa B pelo valor na caixa C
====

Exercício:
Experimente reproduzir os passos acima com outros valores, por exemplo:
 2 x 3
 7 x 5
34 x 3
 1 x 2
 0 x 3
-5 x 4
====

Veja o que acontece com

 4 x -5
Tente explicar por que acontece isso?

====

É possível registrar QUAL linha (instrução) do conjunto de passos (do programa) será executada a cada momento.

Usando uma caixa (apontador) I (índice), podemos, após a leitura de uma instrução, "atualizar" o valor de I para o endereço da "próxima" linha.

Se precisarmos saltar alguma "próxima" linha, preenchemos o registrador I com a "nova" próxima linha.

Mas, atenção, como o registrador I é automaticamente atualizado para a linha que segue a que acabamos de ler, para saltar, devemos sobrescrever em I a linha pulada com a nova linha de destino.

Um marcador de índice assim é chamado geralmente de Contador de Programa (Program Counter - PC) ou, além de outros nomes,

INSTRUCTION POINTER (Apontador de Instruções), ou IP, que será o nome usado neste curso para esse registrador.

Seguindo a proposta acima,
1) verificamos que endereço (linha) o índice I está registrando;
2) recuperamos na lista de instruções o que está escrito no endereço (linha) registrado em I;
3) aumentamos (incrementamos) I de uma unidade, para a próxima procura;
4) realizamos o que está determinado na linha recuperada no passo 2)

ou, dito de outro modo,

Nos passos 1) e 2)

BUSCAMOS A INSTRUÇÃO (NA LISTA, isto é, NA MEMÓRIA)

No passo 3)

ATUALIZAMOS O APONTADOR (para o endereço da próxima instrução sequencialmente)

No passo 4)

EXECUTAMOS A INSTRUÇÃO

Aqui temos o ciclo

BUSCA > ATUALIZA > EXECUTA

que deve repetir-se indefinidamente durante a execução.

====

Uma série consecutiva de jogadas pode ser representada como uma lista de INSTRUÇÕES em um PROGRAMA.

Cada INSTRUÇÃO elementar, pode, CONDICIONAL ou INCONDICIONALMENTE, quando executada:

COPIAR o valor depositado em uma caixa para a outra;

ESCREVER (depositar) certo valor em uma caixa;

INCREMENTAR (aumentar) de 1 o valor de uma caixa;

DECREMENTAR (diminuir) de 1 o valor de uma caixa;

COMPARAR valores entre duas caixas quaisquer, resultando VERDADEIRA OU FALSA alguma das condições A = B, A > B, ou A < B

ADICIONAR ou SUBTRAIR certo valor a uma caixa;

ADICIONAR - ou SUBTRAIR - o valor de duas caixas, deixando o resultado na primeira, na segunda ou em uma terceira caixa;

SALTAR - com ou sem RETORNO - para outro segmento (outra posição qualquer) da lista de instruções

====

As INSTRUÇÕES DE MÁQUINA ou CÓDIGOS DE MÁQUINA podem ser de três tipos:

INSTRUÇÕES DE CONTROLE: que modificam o fluxo sequencial de instruções;

INSTRUÇÕES DE TRANSFERÊNCIA DE DADOS: que "movem" (copiam) dados de um lugar para outro da máquina

INSTRUÇÕES LOGICO-ARITMÉTICAS: que efetuam cálculos lógicos ou aritméticos

====
"Executar" cada uma das instruções acima significa "disparar" um conjunto de processos na máquina capazes de produzir o efeito correspondente a cada instrução.

Como padrões binários podem servir como "interruptores", podemos associar cada instrução a um código executável, a um código de máquina específico.

Uma vez submetido a um registrador apropriado no processador da máquina - no caso, ao REGISTRADOR DE INSTRUÇÕES (RI) - um código de máquina é decodificado e, a seguir, executado.

A execução de uma instrução dispara certas ações (modifica conexões, liga e desliga dispositivos) predeterminadas pela arquitetura da máquina.

O conjunto de instruções de máquina para certa arquitetura de computador, seu Instruction Set específico, é, portanto, codificado (numerado) para "ligar" como interruptores certas ações específicas.

Para serem executadas, as instruções de máquina devem ser "lidas", uma a uma, da MEMÓRIA PRINCIPAL (MP) do computador, isto é, levadas da MP para o RI da máquina, ao final do ciclo que chamamos de BUSCA.

As instruções de máquina são lidas sequencialmente desde a MP, e isto é uma das características que definem um Computador Digital de Programa Armazenado em Memória (CDPAM).

Uma sequência de instruções é montada como um PROGRAMA EXECUTÁVEL para obter certos efeitos e modificações no estado da máquina.

O programa deve ser copiado para a MP de modo que cada um de seus códigos de máquina (instruções) seja sequencialmente recuperado, decodificado e executado pelo processador do computador.

Isto ocorre através do ciclo BUSCA > ATUALIZA > EXECUTA, do seguinte modo.

BUSCA (a instrução na Memória Principal e a transfere para o Registrador de Instruções):

O INSTRUCTION POINTER (IP), registrador especializado do PROCESSADOR, apresenta sempre o ENDEREÇO correspondente à posição (célula) da MEMÓRIA PRINCIPAL que contém a próxima instrução do programa.

Por exemplo, quando um computador é ligado, o IP deve ser preenchido com o endereço da primeira instrução do programa de teste POST (PowerOnSelfTest) e tal endereço copiado no BARRAMENTO DE ENDEREÇOS.

O BARRAMENTO DE ENDEREÇOS (conjunto de conexões) que liga a MP ao PROCESSADOR da máquina permite que os símbolos-interruptores binários ali copiados SELECIONEM certa posição de memória, desligando as outras.

O BARRAMENTO DE DADOS (outro conjunto de conexões) que liga a MP a outra área do PROCESSADOR permite que aquela célula (posição) já selecionada de memória entre em contato com o RI do processador.

Um terceiro conjunto de conexões, chamado BARRAMENTO DE CONTROLE, sinaliza (liga/desliga) a orientação da cópia de dados no sentido MP --> PROCESSADOR (sinal READ, "leitura DA memória").

Desse modo, tratando-se de cópia MP --> PROCESSADOR, o conteúdo da célula "apontada" pelo IP é colocado no BARRAMENTO DE DADOS e copiado para o REGISTRADOR DE INSTRUÇÕES (RI).

Acaba, então o ciclo de BUSCA. O CÓDIGO DE MÁQUINA que estava na célula da MP apontada pelo IP já foi transferido para o RI do PROCESSADOR.

ATUALIZA (o Instruction Pointer):

A PRÓXIMA INSTRUÇÃO a ser buscada na MP para execução é a que reside na(s) CÉLULA(s) contígua(s) SEGUINTE(s) à(s) da instrução que acaba de ser transferida para o processador.

Para buscar uma instrução na MP, a máquina precisa, como vimos, selecionar a célula com a instrução através do endereço da célula no BARRAMENTO DE ENDEREÇOS.

Para a máquina colocar o endereço apropriado no barramento de endereços, o IP deve conter aquele endereço, como vimos.

Como o IP ainda contém o endereço da instrução anterior, ele deve ser ATUALIZADO, isto é, INCREMENTADO, para apontar para a posição da próxima instrução a ser executada.

Em uma máquina de 8 bits, com todas as instruções do mesmo tamanho e com o IP de 1 byte (8 bits), por exemplo, se o IP marcava 1111 1111b (FFh, 255d), deve ser ATUALIZADO para 0000 0000b (ZERO).

Isso porque o padrão binário sucessor de 1111 1111b (=FFh) é ZERO em um registrador (catraca) de 8 bits.

Se o IP marcasse, por exemplo, 1111 1100b (=FCh), seria ATUALIZADO para o sucessor 1111 1101b (=FDh); e assim por diante.

EXECUTA (após decodificar a instrução copiada da MP para o RI no ciclo de BUSCA):

O binário da instrução de máquina capturada para o RI dispara uma sequência de microinstruções na CPU (processador), ligando e desligando conexões, em particular na ALU (Arithmetic Logical Unit).

A ALU permite somar, subtrair ou comparar conteúdos de células da MP ou de REGISTRADORES da CPU, transferindo o resultado para algum GPR (General Purpose Register) do processador e/ou para a MP.

A UNIDADE DE CONTROLE (UC) do processador organiza o seu funcionamento, sinalizações de início e fim dos ciclos e sequências de microinstruções.

MULTIPLICANDO DOIS INTEIROS COM O AUXÍLIO DAS CAIXAS (registradores) A, B, C e Z

registrador A (variável durante a execuçao, também chamada de acumulador - ACC, pois vai "acumulando" valores)

registrador B (valor fixo neste exemplo)

registrador C (serve como contador decrescente - variável de controle)

registrador Z (indicador de resultado zero)

Nota: Z é um registrador-semáforo - ou FLAG.
Z corresponde ao bit ZR do PSW:ProgramStatusWord, com vários bits sinalizadores (zero, vai-um, negativo, direção da leitura, overflow, etc.)

Só utilizaremos o ZR do PSW, isto é, apenas o marcardor "deu zero/não deu zero".
Atenção: o valor "1" no registrador Z significa que "DEU ZERO" é VERDADEIRO. Isto é, que a última operação da máquina DEU ZERO.
O valor "0" no registrador Z significa que "NÃO DEU ZERO", isto é, que é FALSO que a última operação da máquina DEU ZERO.

Uma rotina é uma sequência de instruções com certa finalidade. Uma sub-rotina é uma rotina integrante de outra mais complexa.

Sub-rotina em linguagem de montador (ASSEMBLY) para multiplicar 2 inteiros - etapas:

a) Inicialização das variáveis

Isto é, a colocar valores iniciais nas caixas A, B, C e Z, onde necessário; os "dados" do problema.
Por exemplo, ZERO em A, 7 em B e 3 em C, para multiplicar 7 x 3 com o resultado em A.
Consideramos que a máquina atualiza Z automaticamente com 0 ou 1, respectivamente, se a condição "DEU-ZERO" na última operação é FALSA ou VERDADEIRA.

b) Somar o valor de B com o de A e gravar o resultado em A

Após este passo b), o estado da máquina é:
A = 0 + 7 = 7;
B = 7;
C = 3;
Z = 0;

c) Decrementar C (subtrair 1 do valor em C)

C - 1 = 3 - 1 = 2; ou,
C = C - 1; (C agora contém o valor anterior de C - 1)
(Veja que se a atualização de Z não fosse automática, a cada passo teríamos que atualizar Z: "DEU-ZERO" ou "NÃO"?)

d) Pular para f) se Z indicar condição "DEU-ZERO" VERDADEIRA.

e) Pular para b)

f) Voltar à rotina chamadora externa (RET)

Supondo que houve uma instrução CALL exterior SALTANDO para a instrução no endereço a) desta sub-rotina.

Para RETORNAR para a rotina chamadora, é necessário GUARDAR o endereço de retorno.

Para GUARDAR e RECUPERAR posteriormente o endereço de retorno é necessário saber ONDE foi guardado tal endereço.

O endereço de RETORNO é guardado numa parte da MEMÓRIA PRINCIPAL do computador, chamado STACK, na forma de dados "empilhados".

O último dado colocado vai para o "topo" do STACK.

Os endereços da pilha (STACK) foram numerados na MP no sentido inverso, do maior para o menor. Assim, em hexa:

endereços da MP:

```
  00 <-- começo da MP
  01
  02
  03
  ..
  ..
  ..
  FB
  FC <-- topo atual da pilha contendo algum endereço de retorno de rotina ??????
  FD
  FE
  FF <-- começo da pilha
```

Para saber onde guardamos o último dado ?????? (no caso, no topo da pilha, na célula FC) precisamos "anotar" este endereço FC em algum lugar.

Há um registrador especializado da CPU para isso, o StackPointer (SP).

Quando o SP está marcando FC, sabemos que o topo do STACK tem o endereço FC.

Para recuperar o último dado guardado no STACK, precisamos saber o endereço do topo do STACK.

Sabemos o endereço do topo do STACK na MP recuperando o valor guardado no SP (no caso, FC é o endereço do topo do STACK).

APÓS a LEITURA ("uso", ou "retirada") do dado que estava no endereço FC do STACK, precisamos mudar o valor no registrador SP, que agora terá o valor FD, o endereço do novo topo do STACK.

Por outro lado, para GRAVAR no STACK um novo valor, "acima" do topo, precisamos PRIMEIRO "decrementar" o SP, que passará a valer FB.

Com o SP atualizado para FB, agora podemos gravar no novo topo, sem apagar o que está em FC.

Resumindo:

A execução da instrução CALL XX (XX é um endereço de memória qualquer) compreende:

1) atualizar o endereço do topo do STACK no SP, decrementando (-1) o valor do SP

2) colocar o valor do SP no BARRAMENTO DE ENDEREÇOS selecionando o topo do STACK

3) transferir pelo BARRAMENTO DE DADOS, DO IP PARA A MP, o endereço de retorno que está no IP (endereço da instrução posterior ao desvio)

4) transferir para o IP o valor XX para que a próxima BUSCA de instrução seja feita no endereço XX;

A execução da instrução RET compreende:

1) recuperar o valor do SP (endereço do topo do STACK);

2) colocar o valor do SP no BARRAMENTO DE ENDEREÇOS selecionando o topo do STACK

3) transferir pelo BARRAMENTO DE DADOS, DA MP PARA O IP, o endereço de retorno que está no topo do STACK

4) atualizar o endereço do topo do STACK no SP, incrementando o valor do SP

Exemplo numérico com MNEMÔNICOS substituindo as instruções:

Multiplicar 13 por 7 deixando o resultado no acumulador supondo uma instrução chamadora CALL 00 para a sub-rotina de multiplicação

ATENÇÃO! Lembrar que o flag ZR=1 significa SIM, 'resultado zero' e ZR=0 significa NÃO, 'resultado não-zero'

Conhecendo as funções dos registradores da CPU (IP, RI, PSW-ZR, ACC, etc.),

dos barramentos (ENDEREÇOS, DADOS, CONTROLE) e das células de memória, temos,

no ciclo BUSCA-ATUALIZA-EXECUTA-BUSCA-ATUALIZA-EXECUTA-etc.:

```
...
98 ??
99 CALL 00          ;em algum lugar da MP houve uma chamada CALL para uma rotina com endereço 00, quebrando a sequência 99, 9A,
etc.
9A ????            ;9A é um endereço que deve ser GUARDADO para RETORNO posterior
9B ???             ;se o SP marca FC atualmente, o SP deve ser atualizado para o novo topo, FB; e o valor '9A' GUARDADO na MP no
endereço FB
...                ;ao final (RET) da rotina chamada (CALL 00), o valor do SP (=FB) permitirá recuperar o endereço de RETORNO
'9A'
...
...
...                ;a rotina de multiplicação começa na próxima linha (endereço 00)
00 MOV B, 13       ;carregar B com o valor fixo, no caso, com o valor 13

01 MOV C, 7        ;carregar C com o valor inicial da variável de controle, no caso, com o valor 7

02 MOV A, 0        ;zerar o acumulador

03 ADD A, B        ;somar B com A deixando o resultado em A

04 DEC C           ;subtrair 1 do valor em C

05 JNZ 03          ;voltar ao ínicio do laço da sub-rotina se ainda não foram somadas 7 parcelas 13

06 RET             ;retornar à rotina chamadora externa, já que foram somadas 7 parcelas 13

fim_de_m6_060924.txt
```