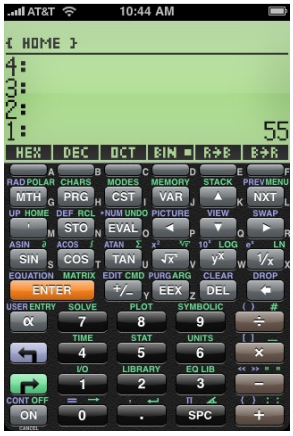
Experimento Bônus

1. Introdução

Construir com 8051 uma calculadora aritmética de até 4 dígitos BCD, operando com a Notação Polonesa Reversa (RPN - Reverse Polish Notation) e pilha de tamanho 4, algo parecido com o que é usado pelas calculadoras HPs 48 e 50. A figura ao lado apresenta uma típica calculadora HP com as 4 posições da pilha explicitadas.



A referência [1] apresenta um simulador que possibilita a prática com a RPN. Informações sobre RPN estão em [2], [3], [4].

2. Descrição da Calculadora

A calculador tem uma pilha com 4 posições que vamos denominar de X, Y, Z e T, como mostrado ao lado. O número teclado sempre aparece na posição X. Isto significa que enquanto se tecla os dígitos do número, eles vão aparecendo na posição X. A sinalização de que se terminou de teclar o número é indicada pela tecla "ENTER↑" ou por uma tecla de operação. As operações sempre acontecem com X ou com X e Y. O exemplo abaixo mostra a operação "30 / 2 = 15". A seqüência de teclas é: **3 0 ↑ 2 /**. Note que ao executar a operação "/" a pilha "desce" e a posição mais "acima", a letra b no caso do exemplo, é repetido.

T
Z
Y
X

Teclas	"partida"	**3**	**0**	**↑**	**2**	**/**
Pilha	d	c	c	b	b	b
	c	b	b	a	a	b
	b	a	a	30	30	a
	a	3	30	-	2	15

3. Problema da entrada e saída

Como este experimento será executado num simulador, é claro que não teremos a disponibilidade de um display e muito menos de um teclado. Portanto, a pilha será construída na RAM interna usando os endereços de 30H até 37H, sendo dois endereços para cada posição da pilha. O byte mais significativo deve ocupar o endereço menor. Ou seja: X=[30,31], Y=[32,33], Z=[34,35], T=[36,37]. Podemos imaginar que existe uma rotina que constantemente escreve essas posições num display.

Cada tecla será Identificada por um código único. Para simular o teclado, vamos também imaginar que existe uma rotina que lê o teclado e coloca o código das teclas acionadas numa fila. Essa fila ocupa as posições de 40H até 5FH, o que resulta num total de até 32 teclas acionadas. A "Tecla inválida" indica o fim desta fila. A presença desta "Tecla inválida" é obrigatória! Em outras palavras, o programa deve ir

processando a fila de teclas até encontrar a tecla com o código "FF", quando então deve parar e entrar num laço infinito.

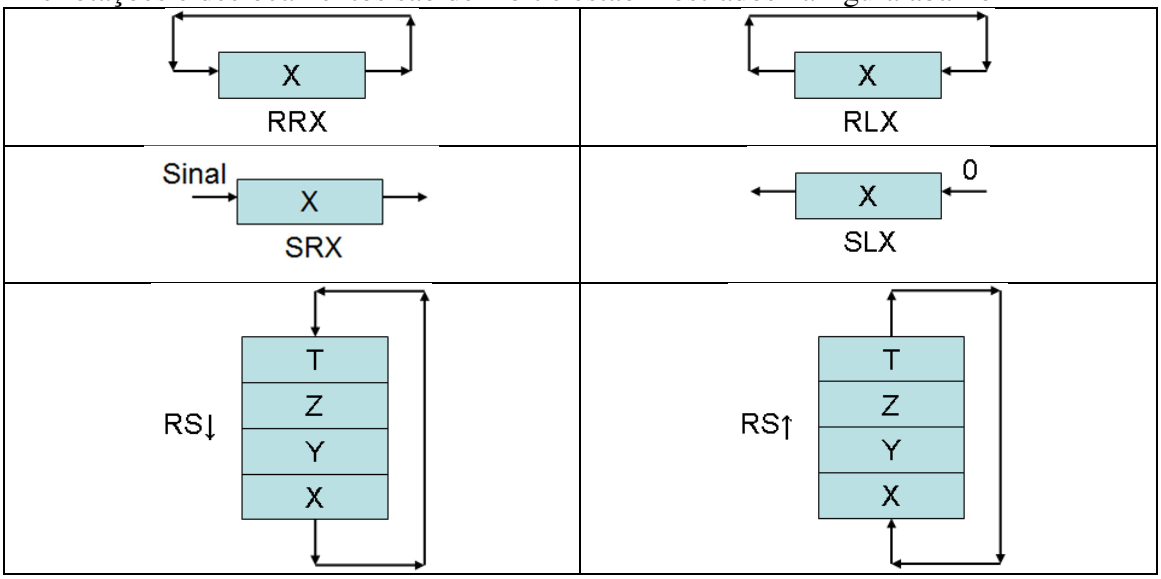
Tabela com os códigos das teclas

Código	Significado
00	Dígito 0
01	Dígito 1
02	Dígito 2
03	Dígito 3
04	Dígito 4
05	Dígito 5
06	Dígito 6
07	Dígito 7
08	Dígito 8
09	Dígito 9
0A	CHS
0B	CLX
0C	CLS
0D	ENTER (↑)
0E	DROP (↓)
0F	SWAP

Código	Significado
10	Somar (Y+X)
11	Subtrair (Y-X)
12	Multiplicar (Y*X)
13	Dividir (Y/X)
14	Módulo (Y/X)
15	RRX (Rodar Direita)
16	RLX (Rodar Esquerda)
17	SRX (Deslocar Direita)
18	SLX (Deslocar Esquerda)
19	X^2
2A	Y^X
2B	RS↑ (rodar pilha ↑)
2C	RS↓ (rodar pilha ↓)
2D	STO (armazena X)
2E	RCL (restaura X)
FF	Tecla inválida

CHS = trocar sinal de X
CLX = zerar X
CLS = zerar toda a pilha
ENTER = push na pilha
DROP = pop na pilha
STO = armazena X em uma memória
RCL = copia memória para X
RS↑ = rodar pilha "para cima"
RS↓ = rodar pilha "para baixo"
SWAP = trocar X e Y

As rotações e deslocamentos são de 1 bit e estão mostrados na figura abaixo



A "memória" para as operações STO e RCL deverá usar as posições [38,39] da RAM Interna.

4. Representação Numérica

A representação utilizada é 4 dígitos BCD, por isso são válidos os números de 0 até 9.999. Considerando o sinal, os possíveis valores vão desde -9.999 até +9.999. As operações são inteiras, com precisão de 4 dígitos BCD. Para representar os sinais dos números que estão na pilha vamos usar 4 bits da posição 20H da RAM interna:

```
20H.0     BIT   SINAL_X    ;bit=0→ positivo, bit=1→ negativo
20H.1     BIT   SINAL_Y    ;bit=0→ positivo, bit=1→ negativo
20H.2     BIT   SINAL_Z    ;bit=0→ positivo, bit=1→ negativo
20H.3     BIT   SINAL_T    ;bit=0→ positivo, bit=1→ negativo
```

5. Condições Anormais

Como condição anormal, consideremos a operação inválida (divisão por zero) e a ultrapassagem (overflow). A ultrapassagem ocorre quando o valor absoluto do resultado de uma operação for maior que 9.999. Ela ocorre também quando o usuário tenta digitar um número de mais de 4 dígitos. Quando ocorrer uma dessas condições, a calculadora sinaliza o problema e para em um loop infinito. Para sinalização serão usados 2 bits da posição 20H.

```
20H.4     BIT   ERRO       ;bit=0→ OK, bit=1→ ERRO
20H.5     BIT   OVFL       ;bit=0→ OK, bit=1→ ultrapassagem
```

6. Detalhes sobre a simulação

Como esta calculadora será apenas simulada, é necessário especificar o ambiente que será usado. Como já foi dito, as teclas serão simuladas por uma fila na RAM Interna, ou seja, antes de iniciar a simulação é necessário preencher esta fila com as operações que se deseja ensaiar. Como não temos display, os resultados serão lidos diretamente na RAM Interna.

6.1. Reserva de espaço na RAM Interna

O programa fica claro se reservarmos espaço na RAM Interna, como mostrado na listagem abaixo. Cada posição da pilha da calculadora é composta por dois bytes, que serão denominados “H” (High) e “L” (Low). Por exemplo, a posição X é composta pelos bytes XH e XL. Serão necessárias posições auxiliares para fazer as contas, por isso foram reservadas as posições de AUX1 e AUX2. A constante F_INI representa o endereço do início da fila com as teclas.

```
XH          EQU         30H   ;HIGH X
XL          EQU         31H   ;LOW X
YH          EQU         32H   ;Y
YL          EQU         33H   ;Y
ZH          EQU         34H   ;Z
ZL          EQU         35H   ;Z
TH          EQU         36H   ;T
TL          EQU         37H   ;T
MEMH        EQU         38H   ;MEMORIA PARA STO E RCL
MEML        EQU         39H   ;MEMORIA PARA STO E RCL
AUX1H       EQU        3AH    ;MSB AUXILIAR 1
AUX1L       EQU        3BH    ;LSB AUXILIAR 1
AUX2H       EQU        3CH    ;MSB AUXILIAR 2
```

```
AUX2L       EQU         3DH   ;LSB AUXILIAR 2
F_INI       EQU         40H   ;Inicio da fila
```

6.2.Códigos das teclas

Facilita muito o trabalho do programador se for dado um “apelido” para cada uma das teclas, algo parecido com a sugestão abaixo.

```
CHS       EQU       0AH  ;Mudar sinal de X
CLX       EQU       0BH  ;Zerar X
CLS       EQU       0CH  ;Zerar a pilha (Clear Stack)
ENTER     EQU       0DH  ;"Enter" (PUSH)
DROP      EQU       0EH  ;POP
SWAP      EQU       0FH  ;Trocar X e Y
MAIS      EQU       10H  ;Y+X
MENOS     EQU       11H  ;Y-X
MULT      EQU       12H  ;Y*X
DIVI      EQU       13H  ;Y/X
MODUL     EQU       14H  ;Módulo de Y/X (Resto da divisão)
RRX       EQU       15H  ;Rodar Direita
RLX       EQU       16H  ;Rodar Esquerda
SRX       EQU       17H  ;Shift Direita
SLX       EQU       18H  ;Shift Esquerda
QUAD      EQU       19H  ;X*X
POW       EQU       1AH  ;Y^X
RSUP      EQU       1BH  ;Rodar pilha para cima
RSDW      EQU       1CH  ;Rodar pilha para baixo
STO       EQU       1DH  ;Armazenar X na memória
RCL       EQU       1EH  ;Copiar a memoria para X
T_INV     EQU       0FFH ;Tecla inválidad
```

6.3.Uso de subrotinas

O programa deve estar estruturado em sub-rotinas. Documente de forma clara o que faz cada uma das sub-rotinas, assim, elas poderão ser usadas com segurança por outras sub-rotinas. Por exemplo, as sub-rotinas PUSH e POP poderão ser usadas nas operações aritméticas.

6.4. Teclas numéricas

A entrada das teclas numéricas é simples, entretanto, os alunos costumam ter dificuldades. A sequência é: multiplicar a posição X por 10 e somar o dígito que acabou de chegar. Repetir até chegar uma tecla não numérica. O algoritmo torna clara a ideia.

1) X=0
2) Repetir enquanto for tecla numérica
    2.1) X = 10 * X
    2.2) X = X + tecla
3) Retornar

6.5. Preparação para simulação

Para testar a calculadora, vamos fazer uma simulação das teclas que foram acionadas. O programa deve carregar essas teclas na fila. Por exemplo, a operação 42 + 354 será simulada pela seguinte arrumação da fila.

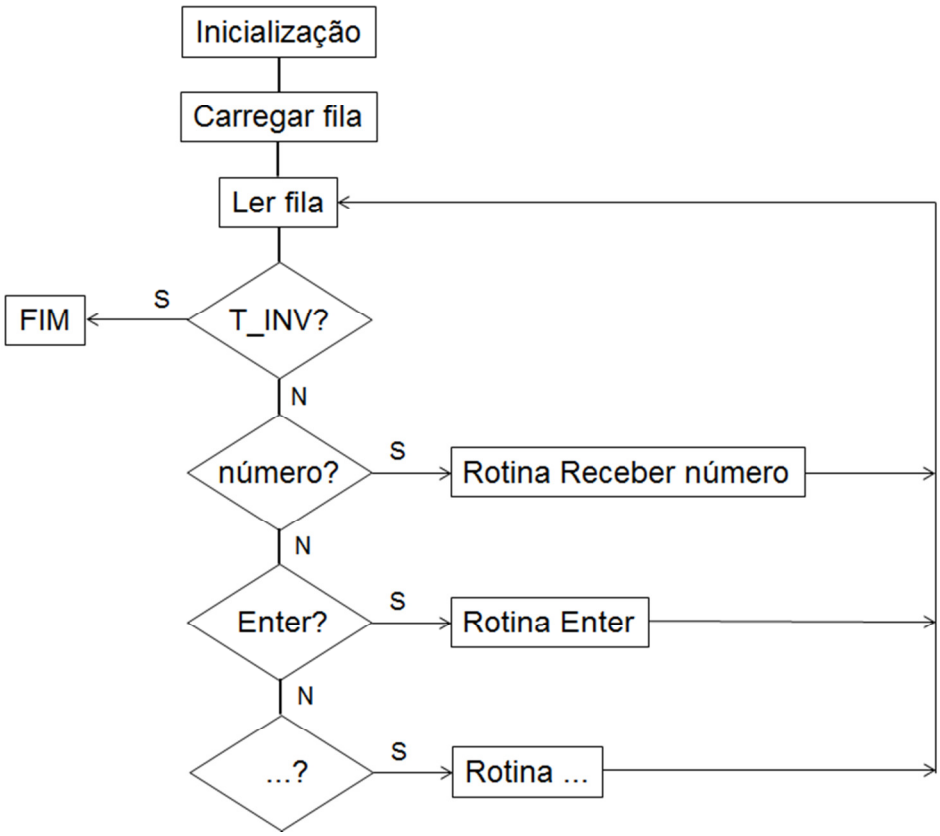
Fila→	40H	41H	42H	43H	44H	45H	46H	47H
Código (Hexa)	04	02	0D	03	05	04	10	FF
Tecla	4	2	Enter	3	5	4	+	T_INV

Para tanto usamos a sequência abaixo e em seguida rodamos o simulador para conferir o resultado.

```
MOV   40H,#04
MOV   41H,#02
MOV   42H,#0DH
MOV   43H,#03
MOV   44H,#05
MOV   45H,#04
MOV   46H,#10H
MOV   47H #0FFH
```

6.6. Organização do programa

A estruturação de um programa assembly é fundamental para que o programador tenha certeza do que está fazendo. O fluxograma abaixo apresenta uma sugestão.



7. Bibliografia

[1] http://www.arachnoid.com/lutusp/calculator.html
[2] http://www.calculator.org/rpn.aspx
[3] http://www.calculator.org/rpn.aspx
[4] http://en.wikipedia.org/wiki/Reverse_Polish_notation