Novo endereço!

Jornal do hpclub do Brasil

http://www.hpclub.com.br

Edição nº 17 – 13/11/2000

Programação sysrpl

As calculadoras HP possuem mais de uma linguagem de programação aceitável, as principais são:

- USER-RPL programas digitados diretamente na HP (<< >>)
- Sys-rpl (System RPL) linguagem usada na programação do sistema da calculadora, necessita compilador (External)
- ML (Machine Language ou Assembler) linguagem de baixo nível Assembler, necessita compilador (Code)

O sysrpl é muito semelhante ao User na maioria de seus comandos e estrutura de programação, a diferença é que com ele você tem uma maior velocidade de execução e maior controle da calculadora e de suas funções.

Não existe nenhum bom livro dessa linguagem de programação em Português, mas você encontra algumas apostilas em inglês sobre essa linguagem na hpcalc.org. A partir desse mês será lançado nos jornais do hpclub do Brasil exemplos de programas em sysrpl começando com exemplos básicos, para que seja fácil seu entendimento até programas mais elaborados com o passar do tempo.

Antes de iniciar a programar em sysrpl, assim como em ML é necessário um compilador. Para HP48GX minha recomendação fica por conta do Jazz (disponível para download no hpclub do Brasil) e para HP49G o próprio MASD encontrado na calculadora (necessita tabela de mnemônicos). Para compilar uma string com um código sysrpl com o Jazz deve-se usar o comando ASS e com o MASD o comando ASM. A diferença básica necessária é que o comando ASM necessita que a string termine com @, como será visto nos exemplos a seguir.

Com o Jazz você pode ter também um Editor de strings, para isso é só entrar uma string vazia na pilha operacional e teclar em ED. Nesse editor você tem alguns atalhos de teclado muito úteis, os principais são:

```
[α] [<-] << >>    - entra com caracteres limitadores do código sysrpl (::  ;)
[α] [->] ::       - entra com limitadores do código ML (CODE  ENDCODE)
[A]   - defini inicio de seleção
[B]   - define final da seleção
[C]   - copia seleção
[D]   - deleta seleção
[F]   - procura por seqüência de caracteres no código digitado
[PRG]       - entra na tabela de comandos
[<-] EDIT   - sai do editor compilando o programa automaticamente
```

Outra coisa interessante com o Jazz é que quando o modo alfanumérico esta acionado, a tecla [ENTER] serve como quebra de linha, facilitando assim a digitação de programas na própria HP48.

Iniciando:

Assim como um programa User é limitado pelos caracteres << e >>, um programa sysrpl é sempre limitado com os caracteres :: e ;. Do mesmo modo que em user, um programa pode conter dentro de si uma rubrotina limitada pelos seus caracteres limitantes (<< e >>), em sysrpl isso é muito comum, só que ao ser executado o programa será tratado de outro modo como pode ser visto no exemplo a seguir:

Exemplo User-rpl

```
<<
   <<
    1.
    2.
   >>
>>
```

ao ser executado retorna na pilha operacional << 1. 2. >>

Exemplo sysrpl

```
::
  ::
   %1
   %2
  ;
;
```

ao ser executado retorna na pilha operacional 1. 2.

Os caracteres :; e ; servirão, além de limitadores de programas, como limitadores de rotinas, como por exemplo com ITE (IF... THEN ... ELSE). Isso poderá ser visto melhor e entendido mais adiante.

Checando objetos na pilha operacional:

Um dos motivos de um código sysrpl ser mais rápido que um código User é que em sysrpl os comandos não fazem checagem de argumentos, ou seja, se no meio de seu programa você especificar uma soma de reais, mas os objetos encontrados forem strings o resultado poderá ser a perda da memória de sua HP! Isso é valido também no inicio de um programa. Devido a isso, é possível facilmente fazer uma checagem de argumentos no inicio de seu programa.

CK1, CK2, CK3, CH4, CH5 - checa se há o número de objetos definidos pelo comando na pilha operacional
CKN - checa se há N objetos definidos por um binário inteiro (#n) na pilha
CK&DISPATCH1 – checa o objeto encontrado na pilha operacional
CK&DISPATCH2 – checa os dois objetos encontrados na pilha operacional.

Para entender melhor esse comando veja o exemplo a seguir:

```
::                * inicia o programa
 CK2              * checa se há dois objetos na pilha operacional
 CK&DISPATCH2     * checa o tipo dos objetos
 #11              * se forem dois reais (tipo 1)
 ::               * inicia sub-rotina
  %+              * executa soma dos reais
 ;                * finaliza subrotina
 #33              * se forem strings (tipo 3)
 ::
  &$              * une as strings
 ;
;                 * finaliza o programa
```

Tacio - hpclub

Você sabia?

- Os valores de pontos vistos no ambiente PICT são diferentes dos valores entrados inicialmente por causa da resolução da tela, quando você plota os pontos, eles são colocados na tela na posição mais próxima do valor real só que há um erro de arredondamento por causa da baixa definição da tela da HP (131 x 64 pixels). Por isso ao olhar um valor pelo gráfico, muitas vezes ele acaba mostrando um valor diferente do real.

- O comando TMENU direciona o menu da HP48/49 diretamente para um menu específico, de acordo com o seu número. Para entender melhor digite 14.03 TMENU, onde 14 é referente ao menu [MTH] |REAL| e .03 a terceira página desse menu. Para mais detalhes veja os endereços dos menus no Guia do Usuário.

Tacio – hpclub

Como unir pontos de um gráfico Scatter (dispersão) na HP49

Um simples programa em USER que pode ser facilmente digitado na sua HP49 para unir os pontos de um gráfico Scatter (dispersão) é descrito abaixo.

```
<<
{ #0h #0h } PVIEW            @ Mostra o ambiente PICT
ΣDAT AXL SORT -> l           @ Converte ΣDAT em lista e ordena seus elementos
  << 1. l SIZE 1. -          @ Inicia LOOP de acordo com o tamanho de ΣDAT
    FOR i l i GET EVAL R->C  @ Retira da lista as coordenadas do gráfico e
l i 1. + GET EVAL R->C LINE  @ liga seus pontos no PICT
    NEXT                     @ Termina o LOOP
PICTURE                      @ Entra no ambiente PICT
  >>
>>
```

Tacio - hpclub

Como retirar de uma lista o maior e o menor objeto

Uma forma melhor de resolver o problema seria a seguinte: escrevemos dois programas, uma para o máximo e outro para o mínimo. Assim a solução fica mais geral. Se precisarmos do máximo e do mínimo, escrevemos um outro programa que chame os dois.

```
LMIN:
        << << MIN >> STREAM >>
LMAX:
        << << MAX >> STREAM >>
LMINMAX:
        << -> l << l LMIN l LMAX >> >>
```

Marcel Campello Ferreira

INPUT FORM em USER

Além de simples entradas de dados, o comando INPUT pode ser usado de um modo mais pessoal, fazendo com que o modo alfanumérico ou algébrico esteja acionado ou que o objeto entrado seja verificado quando for teclado ENTER.

A estrutura básica do INPUT é a seguinte:

```
"String Título" "String objeto default" INPUT
```

Mas você pode ainda:

```
"" "" INPUT                  Input form sem cabeçalho
"TITULO" { "" α }  INPUT     Input form com modo alfanumérico acionado
"TITULO" { "" ALG }  INPUT   Input form com o modo algébrico ativado
"TITULO" { "" V }  INPUT     Input form com modo Verboso, ao entrar um objeto
```

no InputForm ele checa se a estrutura esta correta do objeto, é muito útil quando vai se usar o comando OBJ-> após o INPUT. No caso de entrar, no INPUT por exemplo [{ - + 'kh | \ ele não aceitará retornando mensagem até que seja entrado um objeto em formato "aceitável".

Você pode também usar definições compostas.

"Título" { "0" α ALG V } INPUT – Input form com objeto default 0 e com modo alfanumérico (α), algébrico (ALG) e verboso (V) ativado. Desse modo é possível melhorar a entrada de dados ao utilizar INPUT e evitar futuros erros.

Tacio – hpclub

Como instalar o Meta-Kernel

Para instalares na GX (com cartão na porta 1 ou com ampliação de memória) faça o seguinte:

- na HP digite MERGE1 para que a HP fique com 256KB de memória (os nomes dos arquivos que vou indicar podem variar dependendo do site onde foi baixado o metakernel)- o meu foi no HPCALC se não me engano.

Depois de descompactar o zip vai-te aparecer uma pasta com o nome DISK, e depois Install. Entre em INSTALL. Vão aparecer dois arquivos: RECV.KER e MKRAM.KER. faça o download do RECV.KER para a sua HP48 e em seguida transfira o MKRAM.KER (utilizando XMODEM de preferencia, pois o arquivo ocupa 128Kb, podes utilizar o hyperTerminal private edition). Agora clique em 'MKRAM.KER' e execute o RECV.KER. Ao fim de 10s terá o metakernel instalado.

José Alberto Novais Machado

Alguns comandos do Menu 256 na HP49G

O menu 256 na HP49 – acessado por 256. TMENU - possui dezenas de comandos úteis em programação, a utilidade de alguns desses comandos é descrita abaixo.

H->	converte código hexadecimal em seu objeto inicial
->H	converte objeto em código hexadecimal
->A	verifica o endereço na memória de um objeto ou comando
A->	retorna o objeto ou comando de um determinado endereço
A->H	converte endereço em código hexadecimal
H->A	converte código hexadecimal em endereço
->CD	converte string hexadecimal em objeto Code
CD->	converte objeto Code em string hexadecimal
S->H	converte string em string hexadecimal
H->S	converte string hexadecimal em string
->LST	semelhante ao comando USER ->LIST
->ALG	monta a partir de objetos na pilha um objeto 'algébrico'
->PRG	monta a partir de objetos na pilha um programa
COMP->	abre objeto composto (programa, lista etc) em seus objetos
->RAM	Copia (dump) qualquer objeto da ROM para RAM
SREV	inverte string (posições) "ABC" -> "CBA"
R~SB	converte real em binário inteiro do sistema (sistem binary) e vice versa
SB->R	converte binário do sistema em binário inteiro (#1h) e vice versa
LR~R	alterna entre real e real extenso
S~N	converte string em nome e vice versa ($>ID ou ID>$ em sysrpl)
LC~C	alterna entre complexo e complexo extenso
ASM->	desmonta objetos Code em seu código fonte
BetaTesting	string com nomes (para testes apenas)
CRLIB	cria biblioteca a partir do diretório corrente
MAKESTR	cria uma string a partir de um real
SERIAL	número serial da HP (diferente do número impresso na carcaça)
ASM	compilador sysrpl e ML (precisa da biblioteca de endereços e mnemônicos)
ER	usado pelo comando ASM2 para que na compilação de programas seja comunicado o erro ao utilizador e localiza a linha
->S2	decompilador sysrpl e ML
XLIB~	gera endereço XLIB a partir de dois reais

Tacio Philip Sansonovski
José Alberto Novais Machado

Iniciando programação ML - parte II

II -> Informações Gerais sobre programação ...

1) Antes de começar ...

Primeiramente é bom se acostumar com os compiladores, um excelente compilador é o jazz pena que ele seja muito grande (~70Kb), mas existem versões menores que foram compiladas com o mínimo de recursos, ou seja, não possuem Debuger ou visualizador de textos.

Se você tiver uma GX ou uma G+ aconselho você pegar o Jazz sem o Editor de textos e o Miniwriter, pois juntos não ocupam muita memória, depois além do Jazz e do Miniwriter você precisa também instalar uma Entrie Table. Esta Entrie Table é uma tabela de informações onde estão contidos todos os nomes das instruções que estão na ROM da maquina, é muito útil pois já pensou se fossemos escrever todo um código de maquina em Hexadecimal?

Caso você queira utilizar outro compilador é bom começar com ele, pois para mim foi muito difícil largar o Jazz, tentei utilizar o Masd mas como a forma de programação, mas ainda estou tentando me adaptar a ele. É um pouco diferente da do Jazz não consegui me acostumar. Caso você utilize o Masd algumas coisas no código terão de ser observadas;

Por exemplo:

As instruções do Tipo GOSBVL =SAVPTR e GOSBVL =GETPTRLOOP são tratadas de forma exclusiva pelo compilador: SAVE & LOADRPL -> São as mesmas citadas acima porém só funcionarão no Masd.

Como eu conheço mais o Jazz e sua linguagem se assemelha bastante à programação do GNUTools ou da HPTools prefiro utiliza-lo.

Após ter feito o Upload dos programas e libs para a sua maquina, é bom também arrumar alguns códigos de maquina que estão por ai espalhados na net, na própria Hpcalc.org existem vários códigos na parte de Programming/Misc. Eles o irão ajudar muito à entender a linguagem de maquina; não ligue se você não conseguir entender nada, demora um pouco para aprender a mexer nessa linguagem, ao menos já começará a se acostumar com os códigos.

2) Instruções de teste....

É bom que você já esteja familiarizado com instruções do tipo If ... then ... else ou ao menos conhecer alguma coisa sobre elas. Em Linguagem de maquina essas instruções funcionam de uma maneira um pouco diferente da convencional:

Existem varias instruções que fazem testes, como a de Carry ou mesmo a de comparações com números. As que utilizam de testes de Carry geralmente são utilizadas para fazer loopings cuja contagem é retrograda, pois é mais fácil e mais rápido, exemplo:

```
GOSBVL  =SAVPTR
        LAHEX   08              -> Muda o Valor do registrador A para 06
        LCHEX   05              -> Muda o valor do registrador C para 05
Loop1   A=A-1   B               -> Decrementa o valor de A[B] de 1
        ?A#C    B               -> Testa se A[B] não é igual a C[B]
        GOYES   Loop1           -> Vá para o Loop1 caso A[B] diferente de C[B]
                                -> Senão continue o programa.
        GOVLNG  =GETPTRLOOP
```

O código acima é bem simples para que você entender, o programa só irá parar quando A[B] for igual a C[B]. Observe que eu carreguei o valor de A[B] com 08 e o valor de C[B] com 05; o A[B] será decrementado de 1 até que ele se torne igual a 5, e quando isso acontecer o programa terminará.

Você pode trabalhar com diversos tipos de testes, igual (?A=B fs), desigual (?A#B fs) igual a 0 (?A=0 fs), diferente de 0 (?A#0 fs), maior ou

menor (?A<B fs) (?A>B fs) e maior ou igual a (?A>=B fs) e menor ou igual a (?A<=B fs).

Uma lista de instruções de teste pode ser encontrada no arquivo SASM.DOC que vem junto ao compilador da Hp (HPTools) ou do GNUTools, ele contém informações sobre cada instrução e sobre a velocidade de cada uma.

Além dos testes com nibbles temos também o teste de bits nos registros A e C:

```
?ABIT=0    n -> Bit n em A[W] é igual a 0?
?ABIT=1    n -> Bit n em A[W] é igual a 1?
?CBIT=0    n -> Bit n em C[W] é igual a 0?
?CBIT=1    n -> Bit n em C[W] é igual a 1?
```

Onde o n é o bit a ser testado.

Existe também o teste do ponteiro P:

```
?P=     n  -> P igual a N?
?P#     n  -> P diferente de N?
```

Os bits do Status também podem ser testados:

```
?ST=0   n  -> testa se o bit n do ST é igual a 0
?ST=1   n  -> testa se o bit n do ST é igual a 1
?ST#0   n  -> testa se o bit n do ST é diferente de 0
?ST#1   n  -> testa se o bit n do ST é diferente de 1
```

Os bits do Hardware Status:

```
?XM=0   ->  Testa se o bit do modulo externo está 0
?SB=0   ->  Testa se o Sticky bit está 0
?SR=0   ->  Testa se o bit de serviço de chamada está 0
?MP=0   ->  Testa se o bit do Modulo Fora está 0
```

Todas as instruções acima requerem um GOYES (vá se sim) ou um RTNYES pois elas precisam ir para algum lugar caso algum resultado seja o esperado.

Em uma instrução If normal seria mais ou menos assim:

IF Teste THEN "Clausula Verdadeira" ELSE "Clausula Falsa"

No caso da programação em ML você teria de verificar qual o Teste a ser feito:

```
        ?A=C    B
        GOYES   Verdade
Falso   Aqui vai o código caso
        A[B] diferente de C[B]
        GOTO Cont               <- Vai para -----
Verdade Essa instrução será executada           |
        caso A[B] igual a C[B]                  |
        e continua normalmente o programa...    |
Cont    Código normal  <------------------------
        .
        .
        .
```

Exemplo:

```
* Este código limpa a tela do sistema.
        GOSBVL  =SAVPTR
* Esta instrução aponta em D0 o endereço do display
        GOSBVL  =D0->Row1                       D0=D0+  16
        LCHEX   #87                             ?C#0    B
        A=0     W                               GOYES   loop1
loop1   C=C-1   B                               GOVLNG  =GETPTRLOOP
        DAT0=A  W
```

```
(mesmo código, para Masd:)
        SAVE                                    DAT0=A.W
        GOSBVL  ="D0->Row1"                     D0=D0+  16
        LC      87                              ?C#0    B
        A=0.W                                   GOYES   loop1
*loop1  C=C-1.B                                 LOADRPL
```

3) Loopings e outros...

Os loopings ou voltas são estruturas "if's" com funções de retorno ou de "goto" combinadas, é por meio de testes que decidimos quando o looping termina ou não; em RPL temos seis tipos de loopings: o start ... next, o start ... step, for ... next, for ... step, do ... until ... end, while ... repeat ... end; cada uma com seu delimitador específico.

Como em ML os loopings são praticamente os mesmos, apenas com algumas modificações em suas estruturas.

Podem ser combinadas clausulas como:

1-> O looping normal de contagem cujo valor é definido, geralmente indo de N até 0:

```
        LAHEX   0F
c1      ...
        código  que não utilize o A[B]
        ...
        A=A-1   B
        GONC    c1
```

Caso você queira utilizar o A[A] apenas mova os dados que estão no registro A[W] para a esquerda com o comando ASL W, e retorne a posição normal com o comando ASR W. Um meio de agilizar o processo de mover o limitador W para a esquerda 5 vezes é chamar a subrotina ASLW5; GOSBVL =ASLW5. E para mover os dados para a posição normal basta utilizar o comando reverso: GOSVBL =ASRW5, ele move os dados do limitador W para a direita.

```
        LAHEX   0F  <- Conta de F até 0 (16x)
c1      GOSBVL  =ASLW5
        ...
        código que possa vir a utilizar o A[A]
        ...
        GOSBVL  =ASRW5
        A=A-1   B
        GONC    c1
```

2-> O looping com o sentido reverso, indo de 0 até N

```
        LAHEX   0F -> Conte até F(15)
        LCHEX   00 -> C[B] -> Valor a ser incrementado
c1      ...
        Código que não utilize A[B] e C[B]
        ...
        C=C+1   B -> Incremente em 1
        ?C#A    B -> C[B] diferente de A[B]?
        GOYES   c1 -> Se sim, vá para c1 e continue o looping
        ...        -> Looping terminado, programa continua normalmente...
        Programa normal
        ...
```

Como no programa acima geralmente nós utilizamos muito o C[A] e o A[A] por isso é bom mover os dados em cada um desses registradores para que seja mais fácil de trabalhar; para isso basta adicionar GOSBVL =ASLW5 no inicio do código

que irá fazer o looping; e GOSBVL =ASRW5 no fim do código do looping. Caso queira obter um dos dados do looping sem afeta-lo basta fazer essas mudanças dentro do próprio código:

* Para não haver nenhum problema com o sistema da maquina lembre-se de limpar a Flag -56 (-56CF)

* Este código produz pequenos 'ticks' utilizando o piezo-eletrico embutido na calculadora...

* Para produzir esses pequenos "ticks" basta setar o bit 11 do registrador OUT...

* lembre-se sempre de que deve limpar o bit 11 do registrador OUT toda vez que sair do programa. Senão problemas irão ocorrer.

```
GOSBVL  =SAVPTR
        LAHEX    3FF   <- são 3FF vezes que ele deve ser executado, a cada execução ele deve "tickar" o numero de vezes que estiver nesse valor.
        LCHEX   000
c1
        GOSBVL  =ASLW5 <- Move o registrador A[B] para a esquerda, salvando-o.
        A=C     X
        CSL     W   <-  Esta pequena parte move o registro C[B] para
        CSL     W       salva-lo, pois ele serve como zona de incremento.
        CSL     W       Tem a mesma função do =ASLW5, mas não existe para
        CSL     W       o registrador C[W].
        CSL     W

        LCHEX   800 <-  Bit 11 Setado
        OUT=C   B <- Escreve esse bit no registrador OUT
lp1                     <- Aqui tem-se uma rotina que serve de "timer"
        A=A-1   X       <- Ela é a responsável pelo efeito.
        GONC    Lp1

        C=0     A  <- Bit 11 limpo; Lembre-se sempre de deixa-lo desligado!!!
        OUT=C   A

        GOSBVL  =ASRW5 <- Retoma o valor de A[X] que deve ser comparado
        CSR     W       <- Retoma o valor de C[X]
        CSR     W
        CSR     W
        CSR     W
        CSR     W
        C=C+1   X       <- Incrementa C[X]
        ?C#A    X       <- C[X] é diferente de A[X]?
        GOYES   c1      <- Sim, então retorne ao Looping.
GOSBVL  =GETPTRLOOP
```

Compile o programa e veja o interessante efeito.

Luis Daniel