PUC

Series: Monographs in Computer Science
and Computer Applications

Nº 8/76

# SUSPENSION OF JOBS ON THE IBM - 1130

by

Rubens N. Melo

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
a Marquês de São Vicente, 209 — ZC-20
Rio de Janeiro — Brasil

Series: Monographs in Computer Science
and Computer Applications
Nº 8/76

# SUSPENSION OF JOBS ON THE IBM - 1130*

by

Rubens N. Melo

Series Editor: Sergio E. R. Carvalho

April, 1976

*This work was supported in part by the Brazilian Government Agency FINEP under contract Nº 244/CT.

Copies may be requested from:
Rosane Teles Lins Castilho, Head
Setor de Documentação e Informação
Departamento de Informática - PUC/RJ
R. Marquês de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - BRASIL

ABSTRACT:

A simple technique which permits the suspension of a running program on a IBM-1130 for later resumption is presented. A slight modification in a system routine is also shown which allows the suspension of program which uses floating-point arithmetic. The simplicity of the technique implies some restrictions but even so it has proved to be very useful in at least one installation.

KEY WORDS:

Suspension of Jobs, interrupt, interrupt subroutine, IBM-1130

RESUMO:

Apresentamos uma técnica simples que permite suspender a execução de um programa no IBM-1130 para posterior continuação deixando o sistema disponível neste intervalo. Uma pequena modificação numa ortina do sistema permite esta suspensão ser efetuada à revelia do programador se o seu programa usa aritmética de ponto flutuante. Apesar de se tratar de uma técnica limitada ela foi usada com sucesso em pelo menos uma instalação.

PALAVRAS CHAVE:

Suspensão de Jobs, interrupção, subrotina de interrupção, IBM-1130.

CONTENTS

# 1 INTRODUCTION

The IBM-1130 system is perhaps the sole computer facility encountered in many small installations in Latin America.

As this system is labelled as scientific, typical application involve a small volume of data movement and a generally large amount of CPU work.

Because of its limited capabilities and specially because of its programmed floating-point arithmetic, it is not uncommon to find jobs of hours of duration an a IBM-1130 system.

Many of the installations mentioned above (universities, for example), use the system at least partially as a laboractory for programming training. In many cases these laboratory sessions take no more than one or one and a half blocked hours and sometimes the time lost between runs in these sessions can be considerable. It would be certainly desirable that another long job could be able to resume its execution in an interleaved manner during the laboratory session. A similar situation occurs when a small job with highest priority comes up and there is a long job running.

With this motivation, a very simple subroutine called IRED was developed which permits the operator to stop a running program and save its "instantaneous description" [1] that is, its current instructions, data, and registers in a file $MEM for later resumption. Another program called $DTDM brings back the instantaneous description of a program stored in $MEM and reactivates its execution.

Although there are some limitations on the use of this technique it has proved to be a very useful procedure at the installation of the INSTITUTO TECNOLOGICO DA AERONAUTICA (ITA) where the IBM-1130 system is used in a university environment.

Section 2 of this paper, presents the routine IRED and the program $DTDM. Section 3 shows a slight modification in a system routine FARC as a means of communication between the

user program and IREQ and gives some operational procedures.

Finally in Section 4 some remarks end the paper.

The reader is assumed to be familiar with the IBM-1130 system and specially to have some knowledge about the IBM-1130 Monitor [2] , in order to understand the details of the programs.

## 2 THE SUBROUTINE IRED AND THE PROGRAM $DTDM

In appendix A the listing of subroutine IRED is shown. Basically, this subroutine uses the monitor parameters stored in the supervisor at the beggining of memory. The core size, the DISKZ subroutine address etc... are all taken from fixed positions in memory. These positions are certainly dependent on the MONITOR version (in our case the MONITOR is version V2.M9), but we can expect them not to change in the newer versions.

This subroutine has 2 parts. The first part the entry of which is IREQ , changes the address of the interrupt subroutine which serves the attention caused by pressing the INTERRUPT REQUEST KEY (INT.REQ KEY) in the console. The new address is the address of the second part of subroutine IRED (see the label INT in the listing).

Once the first call to IRED is made, the INT.REQ key by no more serves only to abort the running job (as it is usually programmed) but serves also to save the instantaneous description of the running program in a (previously defined) file called $MEM, (with a STORE DATA command for example). It is important to notice that this saving of the running program only takes place IF THE CONSOLE SWITCHES ARE ALL UP. Afterwards the INT REQ key also serves its normal purpose and the system waits for the next job.

It is also important to notice that IRED has to be called once anyway, and that before pressing the INT REQ key the program must be manually stopped by means of the STOP Key, (this will be explained in the description of $DTDM).

Now the system can be used for other jobs while the first one is saved in file $MEM. Of course the cards in the card reader and forms in the printer relative to the first job must be kept aside until the resumption of this job.

## The Program $DTDM

A small program called $DTDM is shown in appendix B. It essentially uses the same parameters used by IRED and brings the contents of file $MEM to memory. The control of the CPU is transferred to the instruction HALT which served the interrupt of the STOP button when the program was stopped before the action of the IRED subroutine.

## The File $MEM

Both the program $DTDM and subroutine IRED use a file named $MEM. This file serves only to save the instantaneous description of the suspended program. This file must be a permanent file defined by one of the DUP commands which define files in the systems.

For instance, if the memory of the system is 32K, the file $MEM must be defined by:

```
// JOB
// DUP
*STOREDATA      WS      UA      $MEM      102
```

## 3 USE OF THE TECHNIQUE

Unfortunately the simplicity of this technique implies some limitations and care must be taken during its use.

As it was pointed out before, a program must first call the subroutine IRED to be able to be suspended. This can be accomplished either directly by inserting a card at the beginning of the source program (with the statement CALL IRED) or insuring that the program calls another subroutine which in turn calls IRED.

Ideally, there should be no manual intervention by the operator and in some cases the decision to suspend a job comes during its execution.

Maybe a more sophisticated method exists to insure that any program calls IRED nevertheless a very simple method was devised for programs which use floating point arithmetic. The system routine FARC which is used in all operations with floating-point numbers was silightly modified in order to call IRED the first time it is activated (see appendix C).

This simple method was chosen instead of changing the operating system lest we could introduce (more) "bugs" in the appendices.

Therefore the suspension of a running program can be done with the following restrictions.

a) The program must have already called the subroutine IRED whether because it had an explicit CALL IRED or because it has already called a subroutine which calls IRED.

b) If the program uses disk files in the working storage of the monitor (WS), these files can be distroyed by subsequent jobs unless the disk cartridge is also saved.

c) The cards, the listing on the printer or drawing on the plotter etc... must also be saved (and later repositioned for the resumption of the program).

d) Only one program can be suspended at a time.

In summary the following steps are necessary to suspend a running program .

I) STOP the program (Push the STOP buttom)
II) Save the cards, listings, etc.
III) Lift up all console switches
IV) Press the INT RED key
V) Reset the console keys
VI) Push the START buttom

Now the system is free for other jobs.

To bring back the suspended program, the following steps must be taken.

I) Reposition what was saved before the suspension of the program except that the cards must be preceeded by the commands:

```
// JOB
// XEQ $DTDM
```

II) Push the START buttom
III) After a STOP push the START buttom again.

From now on the program will resume its execution.

## 4 CONCLUSION

A simple and limited technique for the suspension of jobs on the IBM 1130 system was presented. The listings of

the programs are directly presented in the appendices and the implementation of the technique is therefore immediate.

Modifications in the MONITOR system were avoided because they were much more difficult and may be less safe.

The system has been used in at least one installation and has proved to be useful.

## APPENDICES

PPENDIX A

```
PAGE    1     BB06  ,A                                                  PPEN

// JOB                                                    BB06  ,A   PR FE

LOG DRIVE    CART SPEC    CART AVAIL   PHY DRIVE
  0000         3000         3000         0000

V2 M09   ACTUAL 32K  CONFIG 32K

// ASM
*LIST
0001     09645600 00001          ENT     IREQ
000E              00002    $CORE EQU     /E
002C              00003    $IREQ EQU     /2C
00E6              00004    $I420 EQU     /E6
01E0              00005    $ZEND EQU     /1E0
0065              00006    $S250 EQU     /65
00F2              00007    DZ000 EQU     /F2
00EE              00008    $DBSY EQU     /EE
01DE              00009    WD    EQU     $ZEND-2
01DF              00010    SECT  EQU     $ZEND-1
0000 0   00F1     00011    DIF   DC      $ZEND-$DBSY-1
0001 0   0000     00012    IREQ  DC      *-*
0002 0   6905     00013          STX   1  R+1
0003 01  6500000B 00014          LDX   L1 INT
0005 00  6D00002C 00015          STX   L1 $IREQ
0007 00  65000000 00016    R     LDX   L1 *-*
0009 01  4C800001 00017          B     I  IREQ
000B 0   0000     00018    INT   DC      *-*
000C 01  7406000B 00019          MDM   L  INT,6
000E 00  C400000E 00020          LD    L  $CORE
0010 0   90EF     00021          S        DIF
0011 0   D018     00022          STO      KMEM
0012 0   081F     00023          XIO      SWIT
0013 0   C021     00024          LD       SW
0014 0   F01F     00025          EOR      FF
0015 01  4C18001A 00026          BZ       MTOD
0017 0   3000     00027    JOB   WAIT
0018 00  440000E6 00028          BSI   L  $I420
001A 0   C014     00029    MTOD  LD       DSAM+1
001B 00  D40001DF 00030          STO   L  SECT
001D 0   C00C     00031          LD       KMEM
001E 00  D40001DE 00032          STO   L  WD
0020 0   C0EA     00033          LD       INT
0021 00  D4000065 00034          STO   L  $S250
0023 0   C808     00035          LDD      PAMID
0024 00  440000F2 00036          BSI   L  DZ000
0026 00  740000EE 00037 Q        MDM   L  $DBSY,0
0028 0   70FD     00038          B        *-3
0029 0   70ED     00039          B        JOB
002A 0   0000     00040    KMEM  DC      *-*
002C     0000     00041          BSS   E  0
002C 0   0001     00042    PAMID DC      /0001
002D 0   01DE     00043          DC       WD
002E 31  1B505500 00044    DSAM  DSA     $MEM
0032     0000     00045          BSS   E  0
0032 1   0035     00046    SWIT  DC      SW
0033 0   3A00     00047          DC       /3A00
0034 0   FFFF     00048    FF    DC      /FFFF
0035 0   0000     00049    SW    DC      *-*
0036              00050          END
```

PPENDIX B

```
PAGE    2     DB06   ,A

   000 OVERFLOW SECTORS SPECIFIED
   000 OVERFLOW SECTORS REQUIRED
   021 SYMBOLS DEFINED
    NO ERROR(S) AND  001 WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// DUP

*DELETE                IREQ
D 26 NAME NOT FOUND IN LET/FLET


*STORE       WS  UA  IREQ
CART ID 3000   DB ADDR  2EC0   DB CNT   0004

// ASM
*LIST
000E                00001    $CORE EQU     /E
01E0                00002    $ZEND EQU     /1E0
0065                00003    $S250 EQU     /65
00F2                00004    DZ000 EQU     /F2
00EE                00005    $DBSY EQU     /EE
01DE                00006    WD    EQU     $ZEND-2
01DF                00007    SECT  EQU     $ZEND-1
0000 0   00F1       00008    DIF   DC      $ZEND-$DBSY-1
0001 0   C013       00009    INIC  LD      D+1
0002 00  D40001DF   00010          STO   L  SECT
0004 0   3000       00011          WAIT
0005 00  C400000E   00012          LD    L  $CORE
0007 0   90F8       00013          S        DIF
0008 0   D001       00014          STO      *+1
0009 00  65000000   00015          LDX   L1 *-*
000B 00  6D0001DE   00016          STX   L1 WD
000D 0   C804       00017          LDD      PARAM
000E 00  440000F2   00018          BSI   L  DZ000
0010 00  4C000066   00019          B     L  $S250+1
0012     0000       00020          BSS   E  0
0012 0   0000       00021    PARAM DC      /0000
0013 0   01DE       00022          DC       WD
0014 31  1B505500   00023    D     DSA      $MEM
0018     0001       00024          END      INIC

   000 OVERFLOW SECTORS SPECIFIED
   000 OVERFLOW SECTORS REQUIRED
   011 SYMBOLS DEFINED
    NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// DUP

*DELETE                $DTOM
D 26 NAME NOT FOUND IN LET/FLET


*STORE       WS  UA  $DTOM
CART ID 3000   DB ADDR  2EC4   DB CNT   0003
```

PENDIX C

```
PAGE    1     1M07  ,A

// JOB                                                    1M07  ,A   EN A FER

LOG DRIVE    CART SPEC   CART AVAIL  PHY DRIVE
   0000          3000         3000       0000
                             1002       0001

V2 M09   ACTUAL 32K  CONFIG 32K

// ASM
*LIST
                        00001          LIBR
0000     060590C0       00002          ENT     FARC
0000 0   7002           00003   FARC   MDX     *+2
0001 00  4C800000       00004          BSC  I  *-*
0003 0   7017           00005          MDX     IREC      MODIFICACAO PARA USO
0004 0   C37D           00006          LD    3 125
0005 01  4C28000B       00007          BSC  L  UNDER,+Z
0007 0   1808           00008          SRA     8
0008 01  4C200010       00009          BSC  L  OVER,Z
000A 0   70F6           00010          MDX     FARC+1
000B 0   10A0           00011   UNDER  SLT     32
000C 0   D37D           00012          STO   3 125
000D 0   DB7E           00013          STD   3 126
000E 0   C018           00014          LD      UNDCD
000F 0   7009           00015          MDX     XY
0010 0   C015           00016   OVER   LD      MAXXP
0011 0   D37D           00017          STO   3 125
0012 0   C37E           00018          LD    3 126
0013 0   4828           00019          BSC     +Z
0014 0   C80F           00020          LDD     MAXN
0015 0   4810           00021          BSC     -
0016 0   C80B           00022   PLUS   LDD     MAXP
0017 0   DB7E           00023          STD   3 126
0018 0   C00F           00024          LD      OVRCD
0019 0   D37A           00025   XY     STO   3 122
001A 0   70E6           00026          MDX     FARC+1
                        00027   *   CHAMADA  A SUBROTINA IREC
001B 30  09645600       00028   IREC   CALL    IREQ
001D 0   C002           00029          LD      NOP
001E 0   D0E4           00030          STO     FARC+3
001F 0   70E4           00031          MDX     FARC+4
0020 0   1000           00032   NOP    NOP
                        00033   *  AREA DE CONSTANTES
0022     0000           00034          BSS  E  0
0022 0   7FFF           00035   MAXP   DC      /7FFF
0023 0   FFFF           00036          DC      /FFFF
0024 0   8000           00037   MAXN   DC      /8000
0025 0   0001           00038          DC      /0001
0026 0   00FF           00039   MAXXP  DC      255
0027 0   0003           00040   UNDCD  DC      3
0028 0   0001           00041   OVRCD  DC      1
002A                    00042          END

   000 OVERFLOW SECTORS SPECIFIED
   000 OVERFLOW SECTORS REQUIRED
   012 SYMBOLS DEFINED
    NO ERROR(S) AND   NO WARNING(S)  FLAGGED IN ABOVE ASSEMBLY

// DUP
```

REFERENCES

[1] WEGNER, P. Programming Languages, Information Structures and Machine Organization. New York, Mc Graw Hill, 1968.

[2] I B M - IBM-1130 Monitor Version 2.