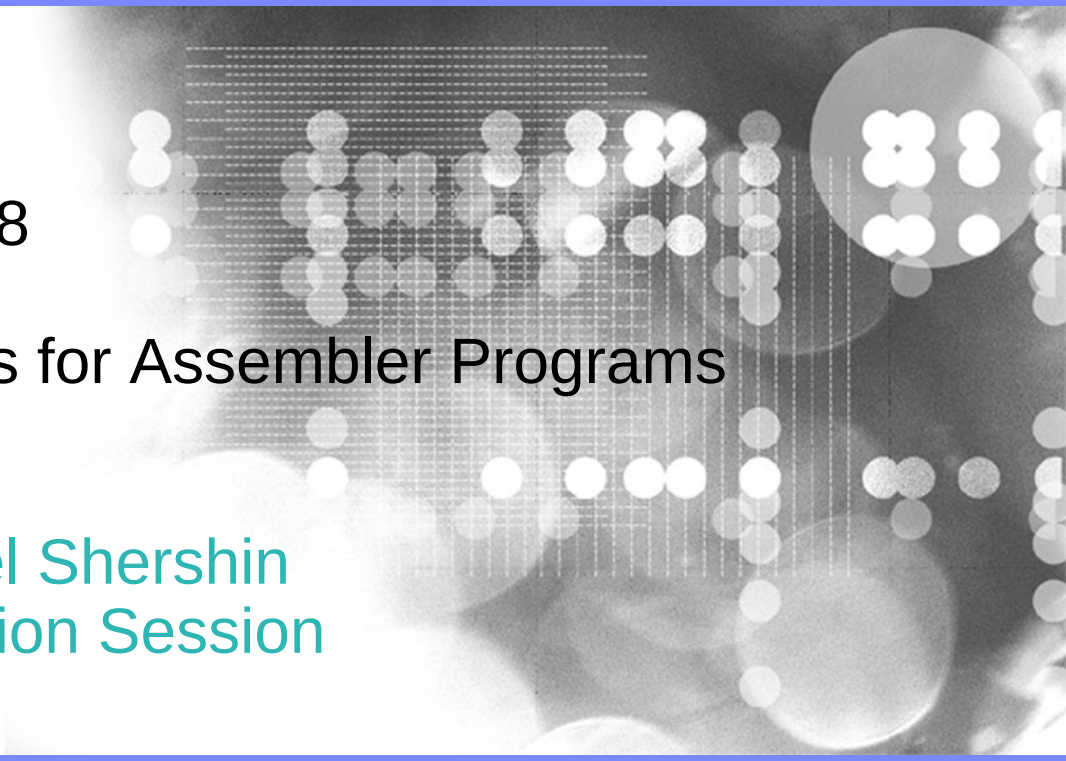
z/TPF V1.1



TPF Users Group Fall 2008

Title: z/TPF Enhancements for Assembler Programs

Name: Michael Shershin
Venue: Education Session

AIM Enterprise Platform Software
IBM z/Transaction Processing Facility Enterprise Edition 1.1.0

Agenda

- **Size greater than 4 K**
  - Multiple base registers
  - Baseless
  - Subroutine support
- **CALLC**
- **Application stack**
- **Register usage**
- **Extended register save option**
- **Storage protect override**
- **Program packaging**
- **New APIs**
- **DECBs**

Size Greater than 4 K

Assembler programs > 4 K

- **No need to split programs**
- **Grow existing assembler programs**
- **Use:**
  - Multiple base registers
  - Baseless support
    - Single base register for first 4K only
      - No base register beyond first 4K
      - Constants in first 4K
    - Baseless for entire program
  - Subroutine support - CLINKC

Multiple Base Registers

- **BEGIN BASE=(Rx, Ry)**
  - Rx is base register for first 4 K
  - Ry is base register for second 4 K
  - Valid registers are R1 – R8

Multiple Base Registers Example

```
            PRINT NOGEN
            BEGIN NAME=TEST,BASE=(R8,R6,R7)
            PRINT GEN
            ENTRC WGR1
+           DS    0H
+           BRASL R14,FIN_WGR1_ENTR   Activate ENTRC stub in FINIS
+           STG   R6,CE3SVR6(R12,R9)                  Save base reg
+           LA    R6,0(R11,R8)                Load second base register
+           STG   R7,CE3SVR7(R12,R9)                  Save base reg
+           LA    R7,0(R11,R6)                Load third base register
            PRINT NOGEN

Note:  R8 set as base register in code generated for BACKC
```

Baseless

- **Assembler programs new routines may not need a base register**
  - Branch instructions can use:
    - Branch Relative on Condition - BRC or J
    - Branch Relative and Save - BRAS or JAS
    - Branch Relative on Condition Long – BRCL
  - Avoid use of literals
    - Use immediate instructions
      - Load halfword immediate – LHI or LGHI
      - Add halfword immediate – AHI or AGHI
      - Compare halfword immediate – CHI or CGHI
      - Multiply halfword immediate – MHI or MGHI
    - Constant handling
      - Use Load Address Relative Long (LARL) to point to constants
      - Put all constants in first 4K of program and use one base register

Baseless

- **Only one base register for program**
  - R8 for initial 4 K only
  - Keep constants in first 4 K
  - Use relative instructions to access data beyond 4 K

Example from cpsi.asm

```
         LARL  R14,CPSITTBL
         TR    CPSICHAR,0(R14)    TRANSLATE PRINT LINE TO
*                                 PRINTABLE CHARACTER.
         ...
CPSITTBL DC    250C'.'          INIT TRANSLATE TABLE TO BLANKS
         ORG   CPSITTBL+C'A'    UPPER CASE A
         DC    C'ABCDEFGHI'
         ORG   CPSITTBL+X'81'   LOWER CASE A
         DC    C'abcdefghi'
```

Baseless

- **Only one base register for program**
  - Use branch relative to go to routines beyond 4 K

```
Example from blog.asm
BLOG0315 DS    0H
         JAS   R1,BLOG0700          CHECK IF GFS ACTIVE
         J     ERROR                RETURN HERE IF NOT
*                                   ELSE GET NEW FA:
         GETFC D8,ID=X'FC33',BLOCK=NO,RTP=9
         ...

BLOG0700 DS    0H
         CINFC W,CMMOPS                  GET ADDRESS OF OPS
         TM    0(R14),X'01'              IS GFS ACTIVE ???
         BOR   R1                        RETURN 0 PAST IF NOT
         B     4(R1)                     RETURN 4 PAST IF ACTIVE
```

Baseless

- **Use no base register in the program**

```
Example from cyma.asm
         BEGIN NAME=CYMA,IBM=YES,BASELESS=YES,BASE=NONE,                +
               TV=CYMB
         ...
*
*              PARSE INPUT MESSAGE
*
          J     CYMA                    ZSONS
          J     CYMB                    ZDMFS
          SPACE ,
CYMA      DS    0H
          ...
          LARL  R2,MSG23H          message text
          LHI   R3,23              message number
WRITE1423 DS    0H
          WTOPC PREFIX=SONS,NUM=(R3),LET=I,TEXTA=(R2),                   x
               COMP=YES,CHAIN=YES,ENDOFM=NO
```

Subroutine Support

- **Support includes the following macros:**
  - CLINKC .. Call subroutine
  - SLINKC .. Start subroutine
  - RLINKC .. Return from subroutine
  - ELINKC .. End subroutine
- **SLINKC saves registers R0 – R13 on entry**
  - R14 is linkage register
  - R15 is pointer to stack
- **RLINKC return restores R0 – R13**
  - One register can be used to return information
  - Multiple RLINKCs can be coded for each SLINKC
  - RLINKC sets R14 and R15 to be linkage address and stack address
    - R14 and R15 can be used within the subroutine
- **ELINKC marks the end of the subroutine**
  - No RLINKCs can be coded after the ELINKC
- **Support limited to the .asm file**
  - Cannot do CLINKC to routine in another .asm file

Subroutine support example from jcs0.asm

```
XC    EBX008(4),EBX008  CLEAR RECORD COUNTER
LA    R2,=C'SPS'    ID
CLINKC RTN=SPX000
L     R1,DC0#SPS    OLD SPS RECORD COUNT
AL    R1,EBX008     PLUS NEW SPS RECORDS
ST    R1,DC0#SPS    NEW SPS RECORD COUNT
SPACE 1
XC    EBX008(4),EBX008  CLEAR RECORD COUNTER
LA    R2,=C'MDS'    ID
CLINKC RTN=MDX000
L     R1,DC0#MDS    OLD MDS RECORD COUNT
AL    R1,EBX008     PLUS NEW MDS RECORDS
ST    R1,DC0#MDS    NEW MDS RECORD COUNT
SPACE 1
XC    EBX008(4),EBX008  CLEAR RECORD COUNTER
LA    R2,=C'MGS'    ID
CLINKC RTN=MGX000
L     R1,DC0#MGS    OLD MGS RECORD COUNT
AL    R1,EBX008     PLUS NEW MGS RECORDS
ST    R1,DC0#MGS    NEW MGS RECORD COUNT
```

Subroutine support example from cpsi.asm

```
CPSI5650 DS    0H
         CLINKC RTN=CPSINDSP           Check for nondisplay storage
...

*-------------------------------------------------------------
*   Subroutine to determine whether any storage areas to be
*   displayed are marked as nondisplayable. When nondisplayable
*   areas are found, they are changed to display as asterisks.
*-------------------------------------------------------------
CPSINDSP SLINKC BASE=NONE
         LA    R6,CPSIPRT              Point to output buffer
...
*************************************************************
*   Return to caller (must be called from SNAPC routine)    *
*************************************************************
CPSNDSP_RET    DS    0H
         AGHI  R6,9                    Increment to next data chunk
         AGHI  R8,4                    Point to next address chunk
         BRCTG R3,CPSNDSP_TOP          Continue three times
         RLINKC ,
         ELINKC ,
```

CALLC

CALLC

- **Ability to call a C function from assembler**
  - Interface for legacy assembler to use C
- **Ability to write a routine once and call regardless of programming language**
- **If a program needs to be split, write new support in C and use CALLC**
- **CPROC**
  - Defines C function prototype to assembler
  - Must have CPROC for each C function used by CALLC

CALLC example from cyc3.asm

Function prototype

```
int tpf_pools_file_in_core_dir(icy7pr * pdsca,
                               icy7sb * buff,
                               TPF_PDIR * dir);
```

Code in cyc3.asm

```
         L     R3,EBW004           LOAD ADDR OF IN-CORE PDSCA  @PJ27469
         L     R4,EBW008           Load pointer to buffer      @PJ27469
         L     R5,EBW000           POINT TO DIRECTORY.         @PJ27469
*************************************************************************
*       KEYPOINT SON POOL SECTION DIRECTORY                             *
*************************************************************************
*
         CPROC RETURN=i,tpf_pools_file_in_core_dir,(p,p,p)
         CALLC tpf_pools_file_in_core_dir(R3,R4,R5)
```

CALLC example from bwra.asm

Function prototype

```
void tpf_recoup_timeout(int restartIndex);
```

Code in bwra.asm

```
*************************************************************
*  Get the restart area index from the ECB work area and call the *
*  Recoup timeout routine.                                  *
*************************************************************
         LLGC  R1,EBW021           PICK UP SLOT INDEX
         CPROC RETURN=void,tpf_recoup_timeout,(i) SETUP LINKAGE
         CALLC tpf_recoup_timeout(R1)    INVOKE FUNCTION
```

Application Stack

Application Stack

- **Available to all programs including assembler**
- **Use of the application stack allows a program to be:**
  - Reentrant
  - Reiterative
    - Variables on the stack rather than in ECB
      - Handle variables in assembler like variables in C
    - Save registers on stack rather than in ECB
- **Two types of usage:**
  - Allocate a fixed amount of space at start of the program
    - Macro CSTKC obtains the pointer to the application stack
  - Dynamically allocate more space on the stack within the program
    - ALLOC in assembler
    - alloca() in C

Application Stack

- **To allocate fixed amount of space at beginning of program**
  - BEGIN DSECT=xxx creates a user defined area on the stack and assigns the specified name to it.
  - BEGIN USEREG=Rx base register for the user stack area.
  - APSTKC – mark end of user defined stack dsect
  - User adds storage definitions between the BEGIN and APSTKC macro.
  - See example on the next slide.

Application Stack Example from crpa.asm

```
         BEGIN NAME=CRPA,IBM=YES,TPFISOC=NO,                           +
               AMODE=64,BASE=NONE,                                     +
               DSECT=CRP,USEREG=R8
C31_TOTAL      DS  D                  Total allocation
C31_BACKED     DS  D                  Currently backed with storage
C31_USED       DS  D                  Currently in use by programs
C31_UNUSED     DS  D                  Area backed but not used
C31_FREE_NUM   DS  D                  Number of blocks on the free chain
C31_FREE_ACUM  DS  D                  Block size accumulator
C31_FREE_SMALL DS  D                  Smallest block on the free chain
C31_FREE_LARGE DS  D                  Largest block on the free chain
C31_FREE_AVRG  DS  D                  Average siz block on the free chain
C64_TOTAL      DS  D                  Total allocation
C64_BACKED     DS  D                  Currently backed with storage
C64_USED       DS  D                  Currently in use by programs
C64_UNUSED     DS  D                  Area backed but not used

         APSTKC
```

Application Stack Example from cvou.asm

```
         ALLOC  SIZE=CVOULEN,ADDR=R4
         USING  REMOVIN,R4
         ...
         L      R2,ISEDUDBI
         UATBC  IDLOC=(R,SSI,R2),NAME=R1,NOTAVL=NEXT1
         ST     R1,CVOUSS2
         ...
REMOVIN  DSECT ,
CVOUPRF  DS    C    PREFIX
CVOUNUM  DS    F
CVOUPGM  DS    CL4   PROGRAM NAME
CVOUSS   DS    CL4   SUBSYSTEM NAME
CVOUSCR  DS    CL8
CVOUSS2  DS    CL4   SUBSYSTEM NAME FROM ENTRY
CVOUFND  DS    CL1
CVOULEN  EQU   *-CVOUPRF
$IS$     CSECT
```

Register Usage

Registers

- **Ability to save / restore registers across an enter to another program**
  - ENTRC PRGM,SAVEREGS=YES
  - Saves R0 – R8 before PRGM is entered.
  - Restores R0 – R8 on return

Example from cvau.asm

```
*-------------------------------------------------------------*
*  Check for user exit requested. At this point the command has passed*
*  all authorization checks. If the user exit is active the customer  *
*  may need to do additional authorization. Call the user exit.       *
*-------------------------------------------------------------*
         TM    FM_AIB,AIB_UEXT     Is user exit active?
         BNO   NO_EXIT             No, continue
         ENTRC UMEX,SAVEREGS=YES    Call user exit
NO_EXIT  DS    0H
```

Registers

- **R10, R11, R12, R13 are available to use in E-type programs**
  - z/TPF only
  - Use as scratch registers
  - Expect that these registers will be destroyed across any macro
    - Extended Register Save Option will save / restore these registers across general macros

Register example from cipx.asm

```
* SEE IF THERE IS AN ACCEPT TASK ACTIVE FOR THIS PROCESSOR
* AND THIS IMAGE
* CWR_FIND_TASK NEEDS TO KNOW WHAT PROCESSOR AND WHAT IMAGE
         LLILH R10,32768             SETUP PROC MASK
         SYSTC SBSDPS,IFOFF=CONTCALL
LCPROC   DS    0H
         CINFC R,CMMPID,F,REG=R6
         PI1DT REG=R6
         LH    R11,PI1CIDN           ARE WE THE FIRST PROCESSOR?
         LTR   R11,R11
         BZ    CONTCALL              YES, SO ALL SET
SHIFTIT  DS    0H                    ELSE BUILD BIT MASK FOR nTH
         SRL   R10,1                 PROCESSOR
         BCT   R11,SHIFTIT
```

Register example from cxck.asm

```
*************************************************************************
*   Set the record to zeroes after the starting with the forward       *
*   chain. The assumption is that the record is a 4K record.           *
*                                                                       *
*       R1 - Base of record to zero.                                    *
*************************************************************************
CLEAR    DS    0H
*
         LA    R14,IPMPPFCH          Point to starting point for zeroing
         LHI   R15,#GBSZE-(IPMPPFCH-ICXPM) Load number of bytes to zero
         SR    R13,R13
         MVCL  R14,R12               Clear out the record.
*
         BR    R6
```

Extended Register Save Option

Extended Register Save Option

- **Ability to save / restore R10, R11, R12, R13 across general macros**
  - Registers will be saved and restored in macro generated inline code.
- **BEGIN EREGSAVE=YES**
  - Activates support in entire program.
- **BEGIN EREGSAVE=ENABLE**
  - Allows support to be activated elsewhere in the program.
  - At start of the program, support is deactivated.
- **DEFBC EREGSAVE=YES**
  - Activates support for a part of the program.
  - Must have coded BEGIN EREGSAVE=ENABLE

Extended Register Save Option Example

```
         BEGIN NAME=TEST,EREGSAVE=YES
         PRINT GEN
         GETFC D7,ID=C'XI',BLOCK=YES   Get pool address
+        LG    R15,PFXECBP3        Get page 3 of the ECB
+        LG    R15,CE3STK2C(,R15)           Get stack frame pointer
+        STMG  R10,R13,ERS.ICST_BR10
+        SVC     GETFC
+        DC      BL.4'0010',AL.4(0)
+        DC      AL1(D7)              DATA LEVEL (REGULAR EXP)
+        DC      AL2(C'XI')           RECORD ID
+        LG    R15,PFXECBP3        Get page 3 of the ECB
+        LG    R15,CE3STK2C(,R15)           Get stack frame pointer
+        LMG   R10,R13,ERR.ICST_BR10
```

Extended Register Save Example

```
          BEGIN NAME=TEST,EREGSAVE=ENABLE
          PRINT GEN
          FINWC D8,ERROR
+         SVC   FINWC
+         DC    AL1(D8)
+         DC    AL1((0*128)+(0*128)+0)
+         JNZ      ERROR            ERROR BRANCH FOR WAITC

          DEFBC EREGSAVE=YES,PUSH
          FINWC D8,ERROR
+         LG    R15,PFXECBP3       Get page 3 of the ECB
+         LG    R15,CE3STK2C(,R15)          Get stack frame pointer
+         STMG  R10,R13,ERS.ICST_BR10
+         SVC   FINWC
+         DC    AL1(D8)
+         DC    AL1((0*128)+(0*128)+0)
+         LG    R15,PFXECBP3       Get page 3 of the ECB
+         LG    R15,CE3STK2C(,R15)          Get stack frame pointer
+         LMG   R10,R13,ERR.ICST_BR10
+         JNZ      ERROR            ERROR BRANCH FOR WAITC
          DEFBC POP=EREGSAVE
```

Storage Protect Override

Storage Protect Override

- **Ability to write into Key 1 and Key 9 storage**
  - Key 1 storage = ECB private area
  - Key 9 protected areas
    - Globals
    - Newly obtained system heap
- **Use when updating protected areas and stack, ECB, or other parts of the private area**
- **APIs that activate storage protect override**
  - GLMOD OVERRIDE=YES
  - KEYCC OVERRIDE=YES
  - STPOC OVERRIDE=ON
- **APIs that disable storage protect override**
  - KEYRC
  - STPOC OVERRIDE=OFF
- **Cannot give up control when storage protect override is active**
  - SERRC E,06401A

Storage Protect Override Example from cxcv.asm

```
SORTREL  DS    0H
*
         ST    R10,WORKRET          Save the return address
         GLMOD OVERRIDE=YES         Override storage protection since
*                                   the sort routine needs to write into
*                                   both the CCA and ALASC block.
         L     R4,EBW012            Load base of ...
*                                             relationship heap storage.
REL      USING ICSAFRM,R4
         MVC   BASE,REL.ICSAROOT    Pass the address of first in chain.
         LA    R14,L'ICSATOLSS      Load length of target lss.
         ST    R14,KEYSIZE          Pass to sort routine.
         LA    R14,ICSATOLSS-ICSANXT Load displacement of target lss.
         ST    R14,KEYDISP          Pass to sort routine.
         BAS   R10,SORTIT
         ...
         MVC   REL.ICSAROOT,BASE    Re-establish the base of the chain.
         KEYRC ,                    Get back to the ecb storage.
```

Program Packaging

Program Packaging

- **Ability to link edit more than one traditional assembler program into one shared object**
  - Less memory needed for the CRPA than if each traditional assembler program is its own shared object
    - Minimum of 28 K needed if one traditional 4 K assembler program is put into its own shared object.
    - Minimum of 32 K needed if two traditional 4 K assembler programs are packaged into one shared object.
  - Reduces linkage overhead
    - ENTRC TYPE=INT can be used for enters within the shared object.

Example from cvaa.asm

```
         ENTRC CVAU,TYPE=INT          GO TO PROCESSOR INFO BIT CHECKS
```

Program packaging

- **To package traditional assembler programs**
  - Add to the .mak file
  - Add ASM_SRC += program_name

Example from cvaa.mak

```
##################################################################
# Segments to be assembled
##################################################################

ASM_SRC := cvaa.asm
ASM_SRC += cvau.asm
ASM_SRC += cvah.asm
ASM_SRC += cvai.asm
ASM_SRC += cvad.asm
ASM_SRC += cvao.asm
```

New APIs

ECB Owners

- **Register the owner of an ECB**
  - Provides information about the purpose of this ECB
  - 32-byte name
    - 8-byte high level qualifier
    - 8-byte mid level qualifier
    - 16-byte low level qualifier
  - Commands such as ZDCOR have an owner name of:
    - High level qualifier = ISMP
    - Mid level qualifier = 5 digit command or ZDCOR in this case
    - Low level qualifier is not used
  - EOWNRC is API to register an ECB with an owner name
  - ECB owner names are given to any 4 K frames and 1 meg frames obtained by this ECB.
  - ZSTAT OWNER displays block usage based on owner name
  - Owner names are included in dumps

ECB Owner (EOWNRC) Example from cvaa.asm

```
CK_SAC    DS    0H                                                  @41
          LA    R7,32               Request 32 bytes of ECB heap
          MALOC SIZE=R7             Get ECB heap
          LTR   R7,R7               Heap obtained?
          BZ    SKIP_EOWNRC         No, branch and skip EOWNRC
          MVC   0(32,R7),SMP_OWNER  Move in owner skelton
          MVI   MID_QUALIFIER-HIGH_QUALIFIER(R7),C'Z'
          MVC   MID_QUALIFIER-HIGH_QUALIFIER+1(4,R7),EBW080 Move in
*                                   secondary action code
          EOWNRC FUNC=SET,OWNER=R7  Set the owner name
          FREEC BLOCK=R7            Release the ECB heap
SKIP_EOWNRC DS 0H


SMP_OWNER        DC   0CL32          SMP Owner name
HIGH_QUALIFIER   DC    CL8'ISMP'     High level qualifier
MID_QUALIFIER    DC    CL8' '        Mid level qualifier
LOW_QUALIFIER    DC    CL16' '       Low level qualifier
```

ECB Heap

- **Default MALOC requests get 31-bit ECB Heap**
- **Ability to get 64-bit ECB Heap**
  - MALOC HEAP=64BIT
  - CALOC HEAP=64BIT
  - RALOC HEAP=64BIT

Example from ctsh.asm

```
        LLGF   R6,=F'32768'              SIZE OF FD LIST TABLE
        MALOC SIZE=R6,HEAP=64BIT         STORAGE FOR FD LIST
        STG    R6,IPWB_COM_MALLOC        SAVE STORAGE POINTER
        LTGR   R6,R6                     TEST FOR 0 FROM MALOC
        JZ     CTSHEFLT                  JUMP TO FAULT CONDITION IF SO
```

ECB Heap

- **Tag an ECB Heap buffer**
  - Ability to associate fixed name with ECB heap address
    - Alternative means of saving the address of a heap buffer
  - To tag an ECB Heap buffer
    - tpf_eheap_tag()
    - EHEAPC FUNC=CREATE,TAG=,ADDR=
  - Locate a tagged ECB Heap buffer
    - tpf_eheap_locate()
    - EHEAPC FUNC=LOCATE,TAG=,ADDR=

ECB Heap Tagging Example from cybd.asm

```
CYBD_FQNAME DS 0H
         LA    R4,256                Set buffer size to 256
         MALOC SIZE=R4               Maloc ECB heap
         XC    0(256,R4),0(R4)       Clear malloc storage
         EHEAPC FUNC=CREATE,TAG=TAGNAME,ADDR=R4,RC=R6
         CGHI  R6,EHEAPC_OK          Did we tag the buffer ok?
         BNE   CYBDER01              No, go process the error


* Get the address of fqname maloc storage and copy into
* ICCFL_SAVE_PARMS
         LA     R2,TAGNAME
         EHEAPC FUNC=LOCATE,TAG=(R2),ADDR=R4
         LTR    R4,R4                 Did we find the buffer?
         BZ     CYBDER01              No, go process the error


TAGNAME  DC    CL32'CCFL.FQNAME'
```

ECB Heap

- **Obtain information about current heap usage**
  - EHEAPC FUNC=INFO
    - C function is mallinfo()
  - Returns information such as:
    - Number of bytes of in use 31-bit pre-allocated heap
    - Number of bytes of available 31-bit pre-allocated heap
    - Number of bytes of in use 31-bit heap excluding the pre-allocated area
    - Number of bytes of available 31-bit heap excluding the pre-allocated area
    - …Plus additional information

Subsecond Timeout

- **EVNTC / ENQC can specify a timeout of less than 1 second**
- **Time units can be specified in …**
  - Milliseconds TIMEINC=MILLI
  - Microseconds TIMEINC=MICRO
  - Nanoseconds TIMEINC=NANO

Subsecond Timeout Example

```
Example from driver iefc\qxsd.asm
         L     R4,=A(63535)
         ENQC  BLOCK=EBX000,WAIT=INVMSG,TIMEINC=MILLI,TIMEOUT=(R4),    X
               TIMEOUTER=TOUTMSG       PRINT 'ECB TIMEOUT' MSG
         ...
*
****  TIME 4095 MICROSECONDS WITH EVNWC ****
*
         MVC   CE1CR5,=X'0001'          SET COUNT TO 1
         EVNTC LEVEL=D5,TYPE=CNT,NAME=N,DUPNAM=DUPMSG,                 X
               STATE=1052,TIMEOUT=4095,TIMEINC=MICRO
         EVNWC LEVEL=D5,TYPE=CNT,ERROR=TIM810A,NFOUND=NFDMSG
         B     INVMSG                   TIMEOUT SHOULD ALWAYS OCCUR
         SPACE 2
TIM810A  DS    0H
```

PNAMC

- **Find**
  - Search application stack to determine whether a given program name is in the stack
- **Modify**
  - Trace name
- **Save**
  - Program name
  - PAT name
  - Caller's name
  - Trace name

PNAMC Examples

Example from cssb.asm

```
*************************************************************
* File (FILNC - so status can be checked with WAITC)
*************************************************************
         MVC   WRKFIL,CSSBFILN     Set FILNC in work area
         STC   R4,WRKFIV           Set requested data level
         PNAMC NEWTRNAME,NAMETYPE=CALLER, Set tagging name to caller   +
               NOTFOUND=CSSB12            Ignore error condition
CSSB12   DS    0H
         BAS   R7,WRKFIL           Go execute FILE macro
         PNAMC NEWTRNAME,NAMETYPE=PROGRAM Restore tagging name
         LA    R0,CSSBEM1          FILE error message text
         WAITC CSSBERR0            Wait for I/O completion
```

Example from rlch.asm

```
         PNAMC NEWTRNAME,FIELD=CURRENT.RLCH_TRC pgm name to Trace name
         FINWC ,DECB=(R1),RLCH18
```

PNAMC Example from cvct.asm

```
*****************************************************************
*  CHECK TO SEE IF THIS CYCLE MESSAGE WAS BUILT BY CPSF.  IF SO, *
*  THEN THE MESSAGE WAS BUILT AS A RESULT OF A ZRIPL MESSAGE.    *
*  SINCE MANY UTILITIES ARE TERMINATED OVER A RE-IPL, IT IS NOT  *
*  NECESSARY TO ISSUE THE WARNING THAT THE UTILITY SHOULD BE     *
*  SHUT OFF.  SINCE CPSF BUILDS A ZCYCL MESSAGE WITH THE BP      *
*  OPTION, LETHAL UTILITIES ARE NOT CHECKED, THE FOLLOWING CODE  *
*  ONLY NEEDS TO BE DONE BEFORE CHECKING FOR NON-LETHAL          *
*  UTILITIES.  SINCE CPSF DOES AN ENTRC TO CVAA TO PROCESS THE   *
*  ZCYCL MESSAGE, THE PROGRAM NESTING CHAIN IS CHECKED TO SEE    *
*  IF CPSF IS ON IT. IF CPSF IS FOUND, IT MEANS THIS             *
*  ZCYCL MESSAGE WAS BUILT BY CPSF AS A RESULT OF A ZRIPL.       *
*****************************************************************
         PNAMC FINDNAME,FIELD=CVCTCPSF, Called by CPSF?                +
               FOUND=CVCT30        Go if yes
*  Fall through if CPSF is not found                                   *
CVCTNIPL DS    0H
         ...

CVCTCPSF DC    C'CPSF'         USED TO CHECK FOR A ZCYCL MSG
*                              BUILT BY CPSF AS A RESULT OF A ZRIPL
```

SWISC IMMEDIATE

- **Switch a program to a specified I-Stream**
  - Start immediately following the SWISC call
  - No need to call a separate program in order to do SWISC
  - Use when cycling through all I-Streams
  - SWISC TYPE=IMMEDIATE
  - swisc_immediate()

SWISC IMMEDIATE Example from cvmn.asm

```
*********************************************************************
*  ONCE WE HAVE THE THRESHOLD VALUE, UDATE THE I-STREAM PREFIX PAGES. *
*********************************************************************
          CINFC R,CMMIST,F,REG=R14    I-STREAM STATUS TABLE
          DCTIST REG=R14              MAP I-STREAM TABLE
          XR    R5,R5                 CLEAR THE REGISTER
          ICM   R5,B'0011',ISTACTIS   GET NUMBER OF I-STREAMS
          DROP  R14
          XR    R0,R0                 CLEAR THE REGISTER
          USING DCTPFX,R0             PREFIXED PAGE
CVMNPFX1  DS    0H
          SWISC TYPE=IMMEDIATE,IS=R5  CHANGE I-STREAM
          ALG   R4,PFX2ILCTR          ADD VALUE TO IL CTR
          STG   R4,PFX2DLTHV          STORE NEW THRESHOLD
          BCT   R5,CVMNPFX1           LOOP FOR EACH I-STREAM
```

New Tape APIs

- **TPUTC**
  - Write data to tape from any storage area
    - Traditional core blocks
    - ECB Heap
    - System Heap
  - Provide pointer(s) to storage addresses being written
    - Can be multiple discontiguous storage addresses
  - Wait for completion – different from TWRTC
  - Write to general tape or real time tape
- **TGETC**
  - Read data from tape to a specified storage area
    - Can read into ECB Heap
  - Provide pointer(s) to storage addresses where data will be read into

TPUTC Example from cpsi.asm

```
ITPGL REG=R6                FOR TPUTC
IMIDAW
...
LARL  R7,CPSIRTA            RTA TAPE
XC    ITPGL(ITPGHLEN+MIDAWLEN),ITPGL
MVC   ITPGNAME,0(R7)
...
LA    R6,EBX000             BUILD ITPGL HERE
STG   R1,ITPGMADR           TRACE BLOCK
ITRAC REG=R5
LG    R5,EBW096
LG    R5,ITRENUM            # trace items
LA    R5,1(R5)              bump
XR    R4,R4                 clear
M     R4,=AL4(ITRSLOTSZ)    times slot size for data length
A     R5,=AL4(SNPMACRO-SNPRID)  plus header
STH   R5,ITPGMCNT           byte count
OI    ITPGMFLG,MIDAWLST     last item
TPUTC DATA=(R6)             Write TRACE block
WAITC CPSI8400              error
```

Additional APIs

- **EBMAXC**
  - Increase ECB allowed maximums for
    - 31-bit ECB Heap
    - 64-bit ECB Heap
    - ECB private area
- **ECBMC**
  - Adjust ECB resource monitor limits for this ECB
- **ERRNOC**
  - Set or retrieve errno
- **FSYSC**
  - Find system heap storage address
  - GSYSC UNIQUE=YES must have been executed in order for FSYSC to find the system heap address

Additional APIs

- **GETCIC**
  - Get information about the code at the core address specified
  - Best to provide the PAT address
- **OWNERC**
  - Get information about physical block usage by owner name
  - Use in conjunction with EOWNRC
- **SYNCC SYNC,WAIT=YES**
  - Control is returned to the ECB only after the global has been updated on all I-Streams in all processors

DECBs

DECBs

- **Similar to ECB data levels**
  - DECBs are dynamically allocated
- **Can be used in C and Assembler**
- **Exist in TPF 4.1 as well as z/TPF**
  - z/TPF's increased ECB Private area (minimum 4 meg) lends itself to more DECB usage.
  - z/TPF's application stack provides an easier place to save the DECB address

DECB example from bofa.asm

```
IDECB REG=R1
LARL  R3,LADECB           Point to Lost address DECB name
LARL  R4,LAID             LA record ID
LA    R5,#L80L8           LA ordinal number
DECBC FUNC=CREATE,DECB=(R1),NAME=(R3)
LA    R14,DECBC_OK        Get valid return code
CR    R14,R15             Did DECBC return a valid DECB?
JNE   BOFAE100            No, jump
IFAC8 REG=R6
LA    R6,IFACLEN          Size of work area
MALOC SIZE=R6             Get temporary work area
LTR   R6,R6               Maloc worked?
JZ    BOFAE100            No, jump
LARL  R10,BCIBM4          Point to record type name
MVC   IFACREC,0(R10)      Move in record type
XC    IFACORD,IFACORD     Zero ordinal number field
ST    R5,IFACORD+4        Save ordinal number
MVI   IFACTYP,IFACFCS     FACS call
```

…continued on next page…

DECB example from bofa.asm (continued)

```
         FAC8C PARMS=(R6)          Get file address
         CLI   IFACRET,IFACNRM     Good return from FAC8C?
         JNE   BOFAE62             No, jump
         MVC   IDECFA,IFACADR
         MVC   IDECRID,0(R4)       Put record ID into DECB
         MVI   IDECRCC,X'01'           "       "
         FINDC ,DECB=(R1)          File the count record
         WAITC BOFAE63
         …
LAID     DC    X'FC35‘
BCIBM4   DC    CL8'#IBMM4   '      face type ibmm4
LADECB   DC    CL16'IBM_LA_CNT_REC'
```

The End

Trademarks

- IBM is a trademark of International Business Machines Corporation in the United States, other countries, or both.
- Notes
- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.