Mainframe Operating Systems “Boot Camp”
From IPL to Running Programs
Part 2

Session #2896
SHARE 112 in Austin, March 2009

Our Agenda for the Week
#2895 - Part 1: The General Purpose Computer and Interrupts
#2896 - Part 2: From IPL to Running Programs
#2897 - Part 3: SVCs and More SVCs
#2898 - Part 4: Program Interrupts (You Want An Exit With That?)
#2899 - Part 5: FLIH: I/O INTERRUPTS
#2894 - Mainframe Operating System Boot Camp: Highlights

“Tell ‘em what you’re gonna tell ‘em”

• What you need to RUN a program
• The Initial Program Load (IPL)
• The Dispatcher
• SVC First Level Interrupt Handlers (FLIHs)
• The Master Scheduler

What you Need to RUN a Program

Just a Few Good Routines and Control Blocks (CBs)

• Initial Program Load (IPL)
• Dispatching
• SVC - FLIH
  • SVC 8 (Loader)
  • SVC 1 (Wait)
• Master Scheduler
What you Need to RUN a Program

Set up the Following Routines and CBs

- PSA ("Low Core")
- The CVT (Memory label = ‘MYCVT’)
- Other Routines & CBs (in SOS-A)
- Dynamic memory area stack (CB100s), Size X’100’
- TCBs & RBs (In dynamic memory areas)

Required Initialization

- X’0’ IPL    PSW
- X’8’ IPL    CCW1
- X’10’ IPL    CCW2
- X’18’ A(MYCVT)
- X’20’ External Old PSW
- X’28’ Supervisor Call Old PSW
- X’38’ Program Check Old PSW
- X’40’ Input/Output Old PSW
- X’48’ CAW
- X’50’ CLOCK
- X’58’ External New PSW
- X’60’ Supervisor Call New PSW
- X’68’ Program Check New PSW
- X’70’ Machine Check New PSW
- X’78’ Input/Output New PSW

Four 64 byte save areas
- One for each, EX, SV, PC, and IO
- 4 F’0’ at ‘TCBWORDS’ - DSECT is IEATCBP

What you Need to RUN a Program

Typical Locations

- X’1200’ MYCVT
- X’1300’ SVC TABLE
- X’1800’ Dispatcher
- X’1C00’ IPL Program
- X’2000’ 5 FLIH (One for each Interrupt type)

What you Need to RUN a Program

Typical Locations

- X’6000’ Location for SVC Modules
  - SVCs 1, 3, and 8 in Instructor Supplied Macro Library: 'WAITS', 'EXIT' & 'LOADERX'
- X’E200’ Master Scheduler (MS)
- X’F000’ Pool of CB100s
What you Need to RUN a Program

**IPL Program**

- The IPL process consists of
  - I/O Phase - loads OS into memory
  - PSW Loading Phase - load current PSW
- At this point it as if you are at the top of the BIF loop

3 Ways to a New Current PSW

- Via an Interrupt
- Via Load PSW Instruction (LPSW)
- Via Initial Program Load (IPL) process

The Initial Program Load (IPL)

**Establish Addressability**

```plaintext
PROGRAM LOGIC:

STEP 1: Establish addressability

STEP 2: Change the clock

STEP 3: Turn on ASSIST-V Trace Facility

STEP 4a: Set protection key to each 2K of memory

STEP 4b: Test ROPSW for SxCS at the expected address

STEP 5: Print CVNRMB value

STEP 6: Init and chain one TCB and RB for each

STEP 7: Transfer control to dispatcher

Establish Addressability

ORG FIRST4K,X'1C00'
INITIAL PROGRAM LOAD:

STEP 1: Establish addressability

USING IPLPGM,12
HELLO? Problem putting it here?

IPLPGM  BALR 12,0
        BCTR 12,0
        "back up" R12
        to A(IPLPGM)
```
The Initial Program Load (IPL)

**Set Storage Key on all Blocks of Memory**

- **SSK - Set Storage Key**
  - SSK X'igighh' X'igaddress' (bytes “ig” are ignored)
  - 8 bits of 'hh' are: 0-3 = key, 4 = fetch protection

**The Initial Program Load (IPL)**

**Set Highest Available Machine Byte**

- **Acquire 4 CBs**
  - **STEP 6: Init and chain one TCB and RB for each NIP/MS and WAIT**
  - GETCB100
  - Allocate a CB100 for this TCB
  - LR 9,10
  - 9 <-- A(TCB)
  - GETCB100
  - 10 <-- A(RB)
The Initial Program Load (IPL)

Fields to Define in the TCBs

<table>
<thead>
<tr>
<th>TCBRB</th>
<th>A(RB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCBTCB</td>
<td>A(Next TCB on the chain or 0)</td>
</tr>
<tr>
<td>TCBBACK</td>
<td>A(Previous TCB on the chain or 0)</td>
</tr>
</tbody>
</table>

The TCB Chain (no RBs yet)

The TCB Chain with RBs

Fields to Define in the RBs

<table>
<thead>
<tr>
<th>RBFLGS3</th>
<th>bit 0 set to 1 for a PRB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>bit 0 set to 0 for an SVRB</td>
</tr>
<tr>
<td>RBOPSW</td>
<td>X'FF0401190F', AL3(MS) for MS RB</td>
</tr>
<tr>
<td></td>
<td>X'FFE601190F', AL3(LEVELABN) for WAIT RB</td>
</tr>
<tr>
<td>RBLINK</td>
<td>Byte 0 is RBWCF and it must be zero</td>
</tr>
<tr>
<td></td>
<td>Bytes 1-3 are address portion:</td>
</tr>
<tr>
<td></td>
<td>for a PRB, this will be the AL3(TCB)</td>
</tr>
</tbody>
</table>

PRB - MS

PRB - Wait

TCBRB

TCBTCB

TCB - MS

TCB - Wait

TCBBACK

TCBGRS

TCBFRS

TCBRB

TCBTCB

TCB - MS

TCB - Wait

TCBBACK

TCBGRS

TCBFRS

TCBRB

TCBTCB

TCB - MS

TCB - Wait

TCBBACK

TCBGRS

TCBFRS
The Initial Program Load (IPL)

Fields to Define in the CVT (MYCVT)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVTTCBP</td>
<td>A('TCB-Words') DSECT = IEATCBP</td>
</tr>
<tr>
<td>PSATOLD</td>
<td>IEATCBP+4 (set by Dispatcher)</td>
</tr>
<tr>
<td>CVTSVCTA</td>
<td>A(SVCTABLE)</td>
</tr>
<tr>
<td>CVTHEAD</td>
<td>A(1ST TCB IN CHAIN) (set by IPL)</td>
</tr>
<tr>
<td>CVTC100H</td>
<td>A(CB100HDR) Header Address for CB100 CBs</td>
</tr>
<tr>
<td>CVT0DS</td>
<td>A(Dispatcher)</td>
</tr>
<tr>
<td>CVTEXIT</td>
<td>SVC 3 An instruction (!)</td>
</tr>
<tr>
<td>CVTBRET</td>
<td>BCR 15,14 An instruction (!)</td>
</tr>
</tbody>
</table>

The TCB Chain with RBs and CVTHEAD

The Initial Program Load (IPL)

IPL Basics Have Been Accomplished

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Set to system mode and transfer control to dispatcher</td>
</tr>
<tr>
<td>IC</td>
<td>Get Key and Fetch Protect Byte</td>
</tr>
<tr>
<td>LA</td>
<td>Get ARegion.</td>
</tr>
<tr>
<td>SSK</td>
<td>Set protection byte for 2K</td>
</tr>
<tr>
<td>MVI</td>
<td>LEVLEN=1,C'S' Indicate system mode</td>
</tr>
<tr>
<td>MVI</td>
<td>TYPFLAG=1,0' Set Type 1 to zero</td>
</tr>
<tr>
<td>MVI</td>
<td>IPLPSW=1,25 Put 'safety net' (X'0119') at A(0)</td>
</tr>
<tr>
<td>DROP</td>
<td>12 END base R12 range</td>
</tr>
<tr>
<td>L</td>
<td>12,CVT0DS Load address of dispatcher from CVT</td>
</tr>
<tr>
<td>BR</td>
<td>12 Transfer control to DISPATCHER</td>
</tr>
<tr>
<td>DROP</td>
<td>3 END CVT addressability</td>
</tr>
</tbody>
</table>
The Dispatcher

**Basic Functions**

- Enter Dispatcher with
  - R12 - A(beginning of Dispatcher)
  - R3 - A(MYCVT)

The Dispatcher

**Basic Functions**

- Dispatcher 'runs' down the chain of TCBs searching each TCB for:
  1. Non-zero A(TCB)
     
     If A(TCB) == 0, then abend
  2. Non-zero A(RB) in TCB
     
     If A(RB) == 0, do Job End Code
  3. If RBWCF == 0, Do Dispatch!

The Dispatcher

**Entry Code**

```
* STEP 1 : Establish addressability of DESPATCH with R12
DISPATCH DS 04  Begin DESPATCH
* STEP 2 : Establish addressability of CVT, TCB and RB
  USING CVT,4 Establish addressability of CVT
  USING TCB,4 Establish addressability of TCB
  USING RB,5 Establish addressability of RB
* STEP 3 : Find task ready to be dispatched
  L 4,CVTHEAD  R4 - ACB(R)
  DOLOP LTR 4,4 Make sure A(TCB) is not zero
  BZ DSPMVR19 Abend if A(TCB) is zero
  L 5,TCBHEAD Get A(RB) associated with RB
  LTR 5,5 If this TCB has been 'killed' go to step 6 and load
  CLI BNEB,K'NO' If RBWCF is zero
  BZ DODISP This task must be dispatched
  L 4,TCBCH Get next TCB in the chain
  DOLOP Repeat the process
  DODISP Work 25
```

The Dispatcher

**Dispatch Code**

```
DODISP DS 04  Perform the Dispatching
  At this point R3 - ACV(T)
  R4 - ACB(R)
  R5 - A(RB)
  L 3,R(3) R3 now points to TCBWORDS
  ST 4,R(3) Store A(TCB) into TCBWORDS+4
  LD 0,TCBFRS Load all
  LD 2,TCBFRS-8 Floating point
  LD 4,TCBFRS-16 registers from
  LD 6,TCBFRS-24 TCBFRS
  MVC LOMPS(K),RBPSR Move PSW of this task into low core
  MVC TCB15PSR(T),TPSER Store time of Dispatcher in TCB
  LM 0,15,TCBFRS Load all GPRs from TCBFRS
  LPSR LOMPSA and Dispatch this task
```
The Dispatcher

*Job End Code*

- Not until we introduce SVC 3 in Part 3
- Why?
  - Our objective is to get a user’s job running
  - Thus we require SVCs 1 & 8 within the Master Scheduler
  - First we discuss SVC-FLIH in greater detail

SVC First Level Interrupt Handler

*Role of the First Level Interrupt Handler*

- Duties
  1. Save the Essence of the Interrupted Program
  2. Direct execution to appropriate module
  3. Direct execution to code to restore Program from Essence

SVC Register Convention of SVC FLIHs

<table>
<thead>
<tr>
<th>Register</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3</td>
<td>A(MYCVT)</td>
</tr>
<tr>
<td>R4</td>
<td>A(TCB)</td>
</tr>
<tr>
<td>R5</td>
<td>A(RB)</td>
</tr>
<tr>
<td>R6</td>
<td>A(SVC Routine/Module)</td>
</tr>
<tr>
<td>R12</td>
<td>Base register, if needed A(dispatcher) for return</td>
</tr>
</tbody>
</table>

SVC First Level Interrupt Handler

*Register Convention of SVC FLIHs*

- GPRs 13, 15, 0, 1
  - Input Parameter registers
- GPRs 15, 0, 1
  - Output/Return Parameter registers
- SVC Routines may use (store into) the area designated by GPR13, but they will not modify GPR13
SVC First Level Interrupt Handler

**SVC Table**
- Pointed to by CVTSVCTA
- Each entry is 8 bytes in length

```
<table>
<thead>
<tr>
<th>AL4(SVC000)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL4(SVC008)</td>
</tr>
<tr>
<td>AL4(SVC255)</td>
</tr>
</tbody>
</table>
```

**SVC First Level Interrupt Handler**

**SVC 1 (Wait)**
- Input Parameters:
  - R0 - Number of events
  - R1 - A(Event Control Block)
- SVC 1 may increment RBWCF in RB of the issuing program
- SVC 1 with SVC 2 uses ECB to control the ‘running’ of a program

**The Code**

```
L 9,CVTSVCTA       Get A(SVC Table)
LH 8,SVCOLD+2     Get SVC Number from old PSW
SLL 8,3           Multiply by 8
LA 8,0(9,8)       Point to table entry

* L 6,0(8)         Get A(module): SVC Table to module, return on R14
* BALR 14,6        RETURNED R15 -> TCBGRS
* ST 15,TCBGRS+(15*4) RETURNED R1 & R0 -> TCBGRS
* STM 0,1,TCBGRS   RETURNED R15, RBWCF to RB
* L 12,CVT0DS      Get A(dispatcher)
BR 12              Go to Dispatcher
```

**SVC 1 (Wait)**
- Suffice it to say:
  - Issue SVC 1 when bit 1 of the ECB is '0' (the issuing program STOPs!)
  - Issue SVC 1 when bit 1 of the ECB is '1' (the issuing program keeps running)
SVC First Level Interrupt Handler

*SVC 8 (Loader)*

- **Input Parameters:**
  - R0 - Load Address
  - R1 - Available Memory

- **Output Parameters:**
  - R0 - Entry Point Address (EPA) of executable load module
  - R1 - Actual size of loaded program
  - R15 - Return Code
    Typical: 0 = OK  4, 8, ... = Not OK

The Master Scheduler

*Entry into MS*

- **Recall:**
  - IPL branches into Dispatcher
  - Dispatcher 'dispatches' (LPSW) MS TCB/RB to begin execution of Master Scheduler at MS-EPA

*Program Logic*

- **STEP 0**
  - Assemble Master Scheduler Resident Data Area (MSDA)
    - A(first 4K user program region)
    - SSK Byte
    - Pad Byte
    - Set MS-ECB (MECB) to X'7F', C'ECB'
**The Master Scheduler**

*Program Logic*

**STEP 1 (EPA)**

- SVC 1 with R0=1 & R1=A(MECB)

**RECALL:**
- if MECB bit 1=0, SVC 1 increments MS-RBWCF by 1 (MS not Dispatchable)
- if MECB bit 1=1, SVC 1 doesn't change MSRBWCF (MS remains Dispatchable)

- WHILE (MS is Dispatchable)
  - Dispatch the MS

---

**The Master Scheduler**

*Operation*

- The MS reads a record from the input file
- If End of File (EOF) is found, SOS has finished
- If there are records in the input file MS will prepare to run the job (first record of each job contains 'Parm Field')

---

**The Master Scheduler**

*Program Logic*

- **STEP 2 & STEP 3**
  - pad (MVCL) 4K region with pad byte
  - issue SSKs to set protect key for two 2K portions of region

- **STEP 4 & STEP 5**
  - move 'Parm Field' into a Parm Area at highest location in the 4K region
  - establish next lower 72 bytes as 18 fullword save area for the user
The Master Scheduler

User Program Region Before SVC 8

- R0 = X'00A000'
- Unused areas in 4K region
- 72 byte Save Area
- X'88' byte Parm. Area with Header
- R1 = Avail Memory from A to Z

Program Logic

- STEP 6
  - R1 = amount of available space remaining in 4K user region
  - R0 = A (start of 4K region)
  - issue SVC 8 to read object module records from input
  - convert them into an executable load module in the region

The Master Scheduler

User Program Region After SVC 8

- R0 = X'00A000'
- Loaded program is here
- Unused areas in 4K region
- 72 byte Save Area
- X'88' byte Parm. Area with Header
- R1 = Length of loaded program (A - X'00A000')
Program Logic

• **STEP 7**
  • If (successful load) obtain and initialize TCB/RB pair
    (just like in IPL)

User PSW

X'FFk501190F', AL3(EPA)

- I/O mask enabled 'FF'
- Protect key no longer '0', now a non-zero value, k, from MSDA field
- State Bit now set to 1 indicating 'Problem State'
- A(EPA) of loaded program (returned in R0 from SVC 8)
- The user program gets dispatched -
  It can issue SVCs and is I/O-External interrupt enabled

User TCB & RB

Dispatching User Program

- Some GPRs in the user TCBGRS field must be set
  - R1 - A(Parm Area)
  - R15 - EPA (must be in R15 AND PSW)
  - R14 - A(CVTEXIT) = A(MYCVT+X'50')
  - R13 - A(OS provided Save Area)
The Master Scheduler

Program Logic (Continued)

- **STEP 8**
  - [SOS] generate job start message
  - Assist-V XPRNT now, SVC 0 later
- **STEP 9**
  - update fields in MSDA for future jobs
  - increment the Current number of Initiators (CURI)

---

The Master Scheduler

TCB Chain with User TCB

- **Dispatching User Program**
  - With three TCBs in the chain, Dispatcher picks first one: MS-TCB
  - To dispatch User-TCB, MS-TCB must be made non-dispatchable
  - To do this:
    - Set MECB to 0
    - Issue SVC 1 against MECB
The Master Scheduler

Program Logic (Continued)

• STEP 12 (bottom of MS Loop)
  • set MECB to 0 (XC)
  • branch to SVC 1 instruction at top of MS Loop

Dispatching User Program

• Issuing SVC 1 with zero MECB makes MS-TCB non-dispatchable
• SVC 1 increments RBWCF (MS RB now waiting on 1 task)
• With MS-TCB non-dispatchable dispatcher now picks User-TCB

The Master Scheduler

TCB Chain with User TCB

Hooray! The Program is Running
Sneak Preview

Part 3: SVCs and More SVCs

- SVC Types: Enabled or Disabled
- Scheduling and Dispatching the SVRB
- SVC 3 - RB ‘Killer’
- The ‘valve’ (SVC 1 & SVC 2)
- SVC 8 - Loader
- SVC 13 - ABEND

Questions

rrannie@cs.niu.edu
m-kozomara@ti.com
www.cs.niu.edu/~rrannie