Robert Rannie — Northern Illinois University - DeKalb, IL
Milica Kozomara — Texas Instruments - Dallas, TX

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

What you Need to RUN a Program

Required Initialization

X'0'	IPL PSW
	Restart New PSW
X'8'	IPL CCW1
	Restart Old PSW
X'10'	IPL CCW2
	A(MYCVT)

X'18'	External Old PSW	
X'20'	Supervisor Call Old PSW	
X'28'	Program Check Old PSW	
X'30'	Machine Check Old PSW	
X'38'	Input/Output Old PSW	
X'40'	Channel Status Word	
X'48'	CAW	A(MYCVT)
X'50'	CLOCK	Trace Info

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

What you Need to RUN a Program

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)

```
* PROGRAM LOGIC:
*                                                                  I
*       STEP 1 :   Establish addressibility                        P
*       STEP 2 :   Change the clock                                L
*       STEP 3 :   Turn on ASSIST-V Trace Facility                 P
*                                                                  G
*                                                                  M
*       STEP 4a:   Set protection key to each 2K of memory
*                                                                  I
*            4b:   Test PCOPSW for S0C5 at the expected address    P
*                  IF (not S0C5 || not at expected location)       L
*                     force 'quick stop'                           P
*                  OTHERWISE,                                      G
*                     obtain the highest available machine byte and M
*                     store it in CVTMZ00
*                                                                  I
*       STEP 4c:   Print CVTMZ00 value                             P
*                                                                  L
*                                                                  P
*       STEP 5 :   Chain CB100 blocks                              G
*                                                                  M
*       STEP 6 :   Init and chain one TCB and RB for each
*                   NIP/MS and WAIT                                I
*       STEP 7 :   Transfer control to dispatcher                  P
```


The Initial Program Load (IPL)

Establish Addressability

```
         ORG   FIRST4K+X'1C00'     INITIAL PROGRAM LOAD:
*--------
*
* STEP 1: Establish addressibility
*
         USING IPLPGM,12           HELLO? Problem putting it here?
*                                     No base register, must be created
IPLPGM   BALR  12,0                Load near location into R12
         BCTR  12,0                'back up' R12
         BCTR  12,0                          to A(IPLPGM)
```

The Initial Program Load (IPL)

Set Storage Key on all Blocks of Memory

```
* STEP 4a: Set protection key to each 2K of memory
*
         SR    2,2               R2 <- A(lowest storage addr) aka F'0'
*                                and protect key into rightmost byte
*
         SSK   2,2              set protection byte of 0(,R2) to X'00'
         XOPC  4                turn off trace for now
         B     POINTHER         jump into the loop
*
IPLOOP   SSK   2,2                 protection byte next block to X'00'
POINTHER LA    2,X'800'(,2)        point R2 to next 2K block
         B     IPLOOP              repeat for all available memory
*
*
*                                NOTE:
*                                The loop 'terminates' as a program
*                                check interrupt (S0C5), at which
*                                point the PCNPSW will BR to AFTERLOP
*
AFTERLOP MVC   PCNPSW+5(3),ADPCFLIH+1    SET instruction address of
*                                        PSNPSW to the appropriate
*                                        1st level interrupt handler
         XOPC  2                turn trace back on
```


The Initial Program Load (IPL)

SSK - Set Storage Key

- SSK X'igigighh' 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

```
* STEP 4b: Test old PSW for S0C5 at the expected address
*          IF (not S0C5 || not at expected location)
*             force 'quick stop'
*          OTHERWISE,
*             obtain the highest available machine byte and
*             store it in CVTMZ00
*
         CLI   PCOPSW+3,X'05'      Test PCOPSW for S0C5
         BNE   IPL0119              If not, force quick stop
*
         LA    3,POINTHER          GET A(POINTHER)
         CLM   3,X'7',PCOPSW+5  Compare to address in old PSW
         BNE   IPL0119             If interrupt was not at SSK instr,
*                                    force quick stop
*                                  SUCCESS? Finish Step 4b
*
         BCTR  2,0                 get highest available machine byte
*
         L     3,76                Get addressability of CVT
         USING CVT,3                  and apply CVT DSECT
*
         ST    2,CVTMZ00           Store the value in CVTMZ00
```


The Initial Program Load (IPL)

Acquire 4 CBs

```
* STEP 6: Init and chain one TCB and RB for each NIP/MS and WAIT
*
*
*                                Allocate a CB100 for this TCB
         GETCB100
*                                GET CB for TCB
         LR    9,10                  9 <-- A(TCB)
*
*                                GET CB to use as RB
         GETCB100
*                                  10 <-- A(RB)
```

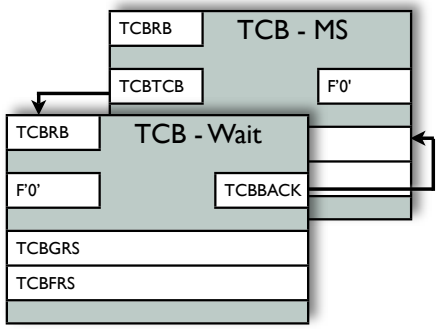
The Initial Program Load (IPL)

Fields to Define in the TCBs

TCBRB	A(RB)
TCBTCB	A(Next TCB on the chain or 0)
TCBBACK	A(Previous TCB on the chain or 0)

The Initial Program Load (IPL)

The TCB Chain (no RBs yet)



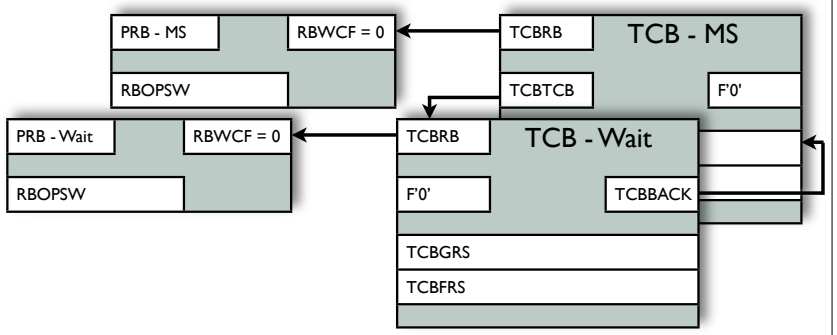
The Initial Program Load (IPL)

Fields to Define in the RBs

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

The Initial Program Load (IPL)

The TCB Chain with RBs



The Initial Program Load (IPL)

Fields to Define in the CVT (MYCVT)

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

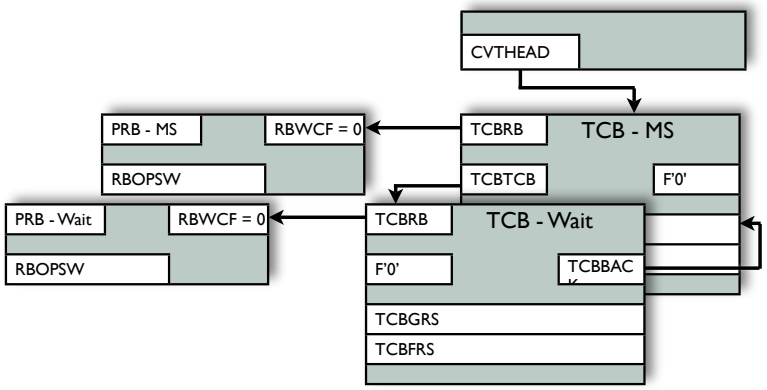
The Initial Program Load (IPL)

Fields to Define in the CVT (MYCVT)

- Build TCB chain for Dispatching
  - Point CVTHEAD to MS-TCB
- Chain TCBs
  - Down from MS-TCB via TCBTCB (lowest TCBTCB = 0)
  - Up from Wait-TCB via TCBBACK (highest TCBBACK = 0)

The Initial Program Load (IPL)

The TCB Chain with RBs and CVTHEAD



The Initial Program Load (IPL)

IPL Basics Have Been Accomplished

```
* STEP 7: Set to system mode and transfer control to dispatcher
*
         IC    11,IPLUCBKE          Get Key and Fetch Protect Byte
         LA    8,X'800'             Get A(region_
         SSK   11,8                 Set protection byte for 2K
*
         MVI   LEVELFLG,C'S'        Indicate system mode
         MVI   TYP1FLAG,C'0'        Set Type 1 to zero
         MVI   IPLPSW+1,25          Put 'safety net' (X'0119') at A(0)
*
         DROP  12                   END base R12 range
         L     12,CVT0DS            Load address of dispatcher from CVT
         BR    12                   Transfer control to DISPATCHer
         DROP  3                    END CVT addressability
```

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 DISPATCH with R12
*
DISPATCH DS    0H                 Begin DISPATCH
         USING DISPATCH,12        Establish addressability
*
*  STEP 2 : Establish addressability of CVT, TCB and RB
*
         USING CVT,3              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 - A(TCB)
DOLOOP   LTR   4,4                Make sure A(TCB) is not zero
         BZ    DISP0119                Abend if A(TCB) is zero
*
         L     5,TCBRB            Get A(associated RB)
         LTR   5,5                If this TCBs RB has been 'killed'
         BZ    DJOBEND                 go to job end code
*
         CLI   RBWCF,X'00'        If RBWCF is zero
         BZ    DODISP                This task must be dispatched
*                                 Otherwise,
         L     4,TCBTCB               Get next TCB in the chain
         B     DOLOOP                  and Repeat the process
*********@*****@*****
DISP0119 XOPC  25                 Assist-V: Force Abend to end simulation
```


The Dispatcher

Dispatch Code

```
DODISP   DS    0H                 Perform the Dispatching
*        At this point R3 - A(CVT)
*                       R4 - A(TCB)
*                       R5 - A(RB)
*
         L     3,0(0,3)           R3 now points to TCBWORDS
         ST    4,4(0,3)           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   LOWPSW(8),RBOPSW   Move PSW of this task into low core
         MVC   TCBTDISP(4),TIMER  Store time of Dispatcher in TCB
*
         LM    0,15,TCBGRS        Load all GPRs from TCBGRS
         LPSW  LOWPSW             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 First Level Interrupt Handler

Register Convention of SVC FLIHs

R3	A(MYCVT)
R4	A(TCB)
R5	A(RB)
R6	A(SVC Routine/Module)
R12	Base register, if needed A(dispatcher) for return

SVC First Level Interrupt Handler

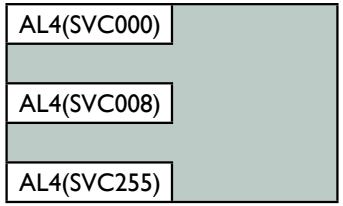
Register Convention of SVC FLIHs

- 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



SVC First Level Interrupt Handler

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


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

SVC First Level Interrupt Handler

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

SVC First Level Interrupt Handler

SVC 8 (Loader)

- 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

The Master Scheduler

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

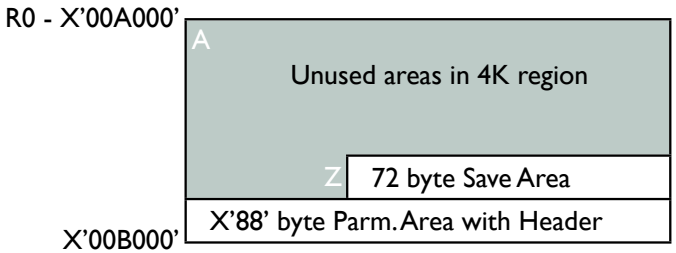
The Master Scheduler

Program Logic

- 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



R1 - Avail Memory from A to Z

The Master Scheduler

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

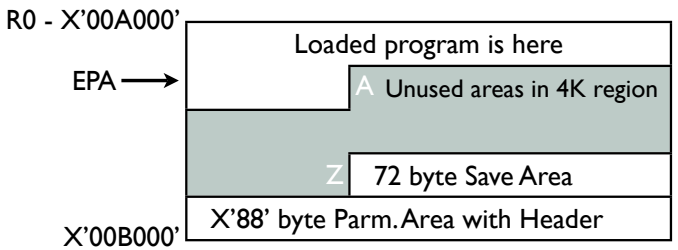
The Master Scheduler

Program Logic

- Upon return from the SVC 8
  - test RC (in R15) for successful load
  - retrieve Entry Point Address of the loaded program from R0

The Master Scheduler

User Program Region After SVC 8



R1 - Length of loaded program (A - X'00A000')

The Master Scheduler

Program Logic

- STEP 7
  - If (successful load) obtain and initialize TCB/RB pair

    (just like in IPL)

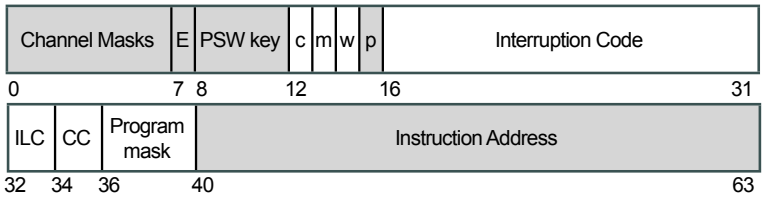
The Master Scheduler

User PSW

X'FFk501190F', AL3(EPA)

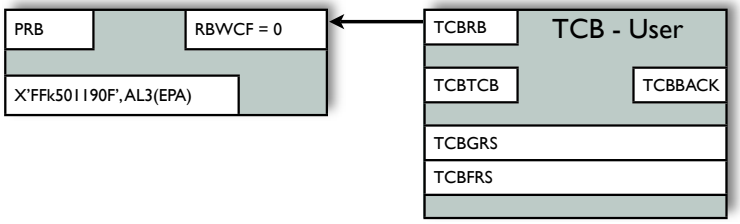
- I/O mask enabled 'FF'
- Protect key no longer '0', now a non-zero value, k, from MSDA field
- State Bit now set to 1 indicating 'Problem State'
- A(EPA) of loaded program (returned in R0 from SVC 8)
- The user program gets dispatched -

  It can issue SVCs and is I/O-External interrupt enabled



The Master Scheduler

User TCB & RB



The Master Scheduler

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)

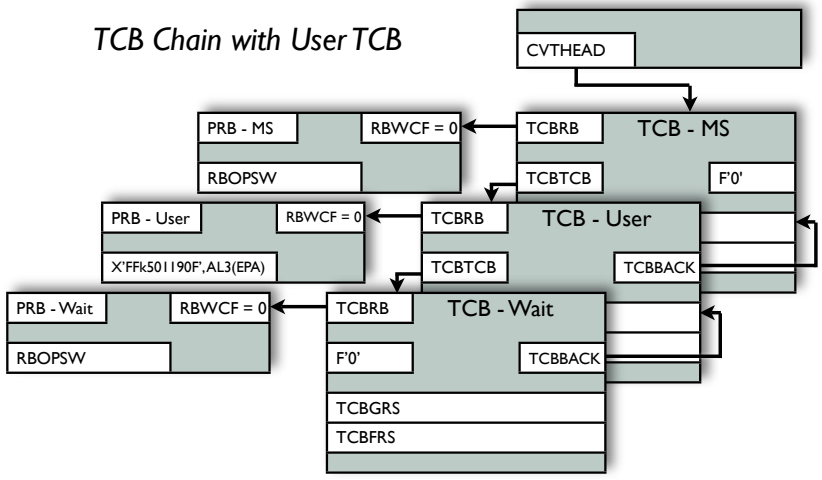
The Master Scheduler

Program Logic (Continued)

- STEP 10
  - Chain user TCB just after MS-TCB
- STEP 11
  - Not until we get to multiprogramming

The Master Scheduler

TCB Chain with User TCB



The Master Scheduler

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

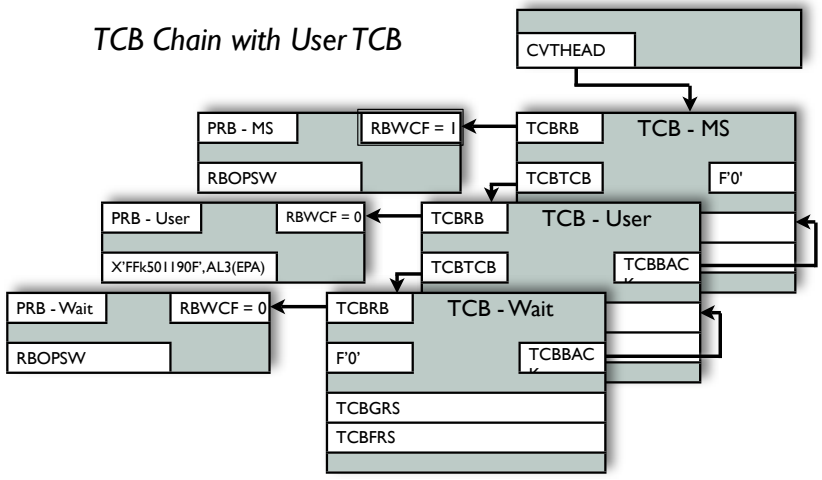
The Master Scheduler

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



The Master Scheduler

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