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

Mainframe Operating Systems "Boot Camp"

FLIH: I/O INTERRUPTS

Part 5

Session #2899
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"

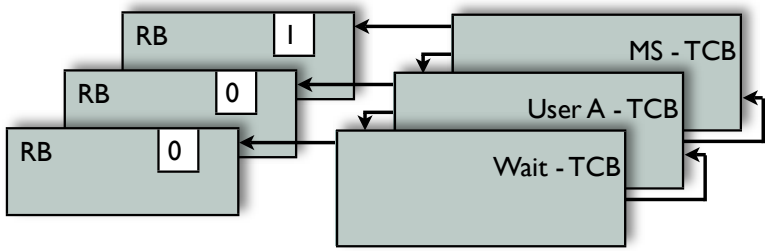
- Dispatch Priority + Wait TCB's Role
- The I/O Concept
- Programs Execute CPU Instructions
- For I/O: New Control Blocks (CB)
- To Start I/O: New Procedures
- I/O Interrupt Processing - I/O FLIH

Dispatch Priority + Wait TCB's Role

- Recall:
  - TCB/RB chain and RBWCF when user job is running
  - Master Scheduler (MS) called SVC 1 to make itself Non-Diapatchable: RBWCF=1
  - The Dispatcher "Dispatched" the next TCB/RB in the chain (Job A)

Job A is now running and wants to do I/O

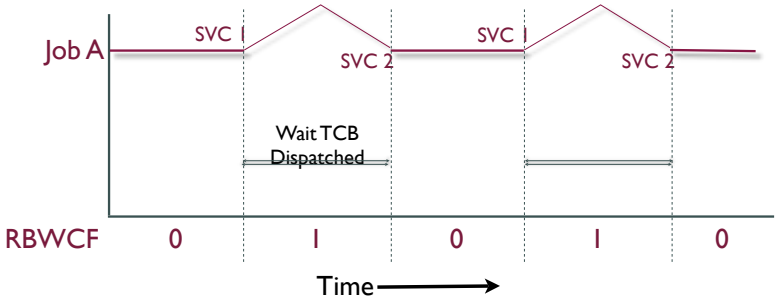
Dispatch Priority + Wait TCB's Role



Dispatch Priority + Wait TCB's Role

- To do I/O:
  - Job A issues SVC 0
  - SVC 0 issues SVC 1 on behalf of Job A:
    Job A is made non Dispatchable
  - After I/O op, IO-FLIH issues SVC 2 on behalf of Job A:
    Job A is Dispatchable again

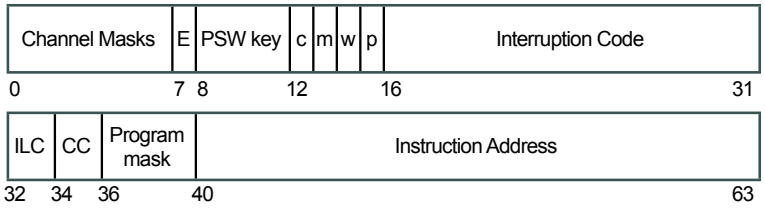
Dispatch Priority + Wait TCB's Role



Dispatch Priority + Wait TCB's Role

PSW in the WAIT RB

X'FFE60I190F',AL3(LEVELABN)



Dispatch Priority + Wait TCB's Role

- Wait TCB/RB :
  - always dispatchable!
  - dispatched when there are no other TCB/RBs
  - Wait Bit set to 1 = Don't Run (i.e. "wait")

Dispatch Priority + Wait TCB's Role

- For the state of the CPU to be changed from Wait, it is necessary for an Interrupt to occur that replaces the current PSW with one in which the Wait Bit = 0
- Optimization will minimize the amount of "Waiting" time

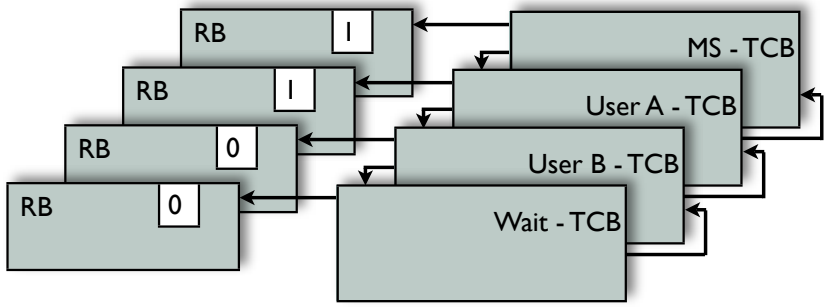
Dispatch Priority + Wait TCB's Role

Example

- Consider Job B in TCB/RB chain below Job A and above Wait
- When Job A starts its I/O, Job B is dispatched

Dispatch Priority + Wait TCB's Role

Example



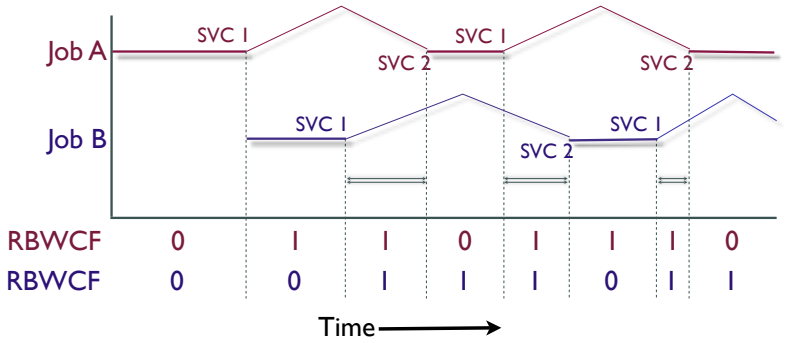
Dispatch Priority + Wait TCB's Role

Example

- Suppose Job B does an I/O by issuing an SVC 0
- SVC 0 issues an SVC 1 on behalf of Job B
- Job B is now non-dispatchable
- After I/O operation, IO-FLIH issues SVC 2 on behalf of Job B
- Now Job B is dispatchable again

Dispatch Priority + Wait TCB's Role

Example



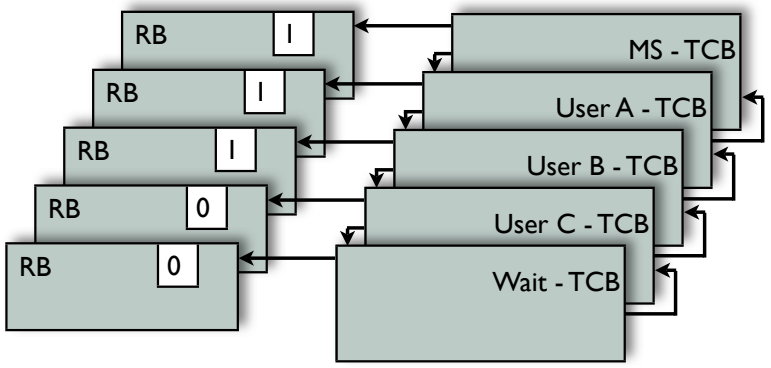
Dispatch Priority + Wait TCB's Role

Example

- Consider Job C in TCB/RB chain below Job B and above Wait
- When Job B starts its I/O, Job C is dispatched

Dispatch Priority + Wait TCB's Role

Example



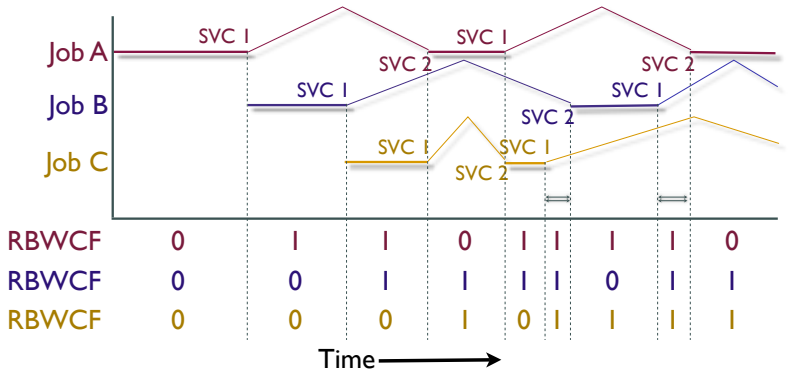
Dispatch Priority + Wait TCB's Role

Example

- Suppose Job C does an I/O by issuing an SVC 0
- SVC 0 issues an SVC 1 on behalf of Job C
- Job C is now non-dispatchable
- After I/O operation, IO-FLIH issues SVC 2 on behalf of Job C
- Now Job C is dispatchable again

Dispatch Priority + Wait TCB's Role

Example



The I/O Concept

Three Steps, and Consideration

- I/O will be Started by the CPU
- Once Started, I/O is Controlled and Processed by the Channel
- Interrupt(s) Signaling Completion are processed by I/O-FLIH
- Should the problem program be allowed to execute while I/O is being done on its behalf?

The I/O Concept

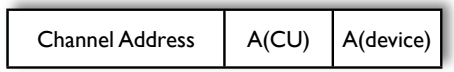
I/O will be Started by the CPU

- User may request I/O by issuing SVC 0
- SVC 0 will issue SVC 1 on behalf of the user
- User program is non-dispatchable until I/O has completed
- Before issuing the CPU instruction to start the CP store A(1st CCW of CP) in Channel Address Word (CAW)
- CAW is located in low core at X'48'

The I/O Concept

I/O will be Started by the CPU

- The hardware address of the specified device must be known
- Device Addresses are given in a two byte form



- Up to 16 Devices may be attached to one Control Unit (CU)
- Up to 16 CU may be attached to one Channel

The I/O Concept

I/O will be Started by the CPU

- The Start I/O (SIO) instruction
  - issued by the CPU to direct the CP to take over I/O
  - specifies the Device Address to be involved in I/O
  - is a privileged instruction
  - will set one of four Condition Codes (CCs)

The I/O Concept

I/O will be Started by the CPU

- The Start I/O (SIO) instruction
  - the original IBM sys/360/370 I/O initiating instruction
  - used in SOS/Assist-v
  - replaced by Start Subchannel (SSCH) in MVS/zOS

The I/O Concept

I/O will be Started by the CPU

- Prior to issuing the SIO instruction
  - obtain the hardware address (Channel, Control Unit, Device) and make it available to SIO
  - build the CP in processor memory
  - store A(1st CCW of CP) in the CAW (at X'48')

The I/O Concept

SIO will set one of four Condition Codes (CCs)

CC = 0

- The Channel has taken control of the I/O operation
- The Channel will execute the CCWs of the CP
- The Channel will continue processing until the last CCW of the CP is encountered or until an error is detected
- At this point the CPU is free to proceed to the next instruction

The I/O Concept

SIO will set one of four Condition Codes (CCs)

CC = 1

- A CSW has been stored (at location X'40')
- The CSW may have information that explains why the Channel chose not to carry out the requested I/O operation

Note: Any of the following three CC indicate that the Channel has Not taken control of the I/O operation. No I/O operation has been started.

The I/O Concept

SIO will set one of four Condition Codes (CCs)

CC = 2

- The Channel was already busy processing an I/O request against the specified device

The I/O Concept

SIO will set one of four Condition Codes (CCs)

CC = 3

- No device was found at the specified device address

The I/O Concept

SIO will set one of four Condition Codes (CCs)

Note: In the SOS, any CC other than zero calls for termination of the program and determination of the error cause.

The possible exception to this practice might be when CC=1 with the Unit Exception bit set to 1 in the CSW indicating End of File (EOF).

The I/O Concept

Started I/O is Controlled and Processed by the Channel

- I/O is started by the CPU and carried out by the channel
- Channel has instructions called Channel Command Words (CCWs)
- Each CCW
  - is eight bytes in length
  - must reside on a double word boundary
  - must reside in processor memory

The I/O Concept

Started I/O is Controlled and Processed by the Channel

- A Channel Program (CP) typically consists of a number of CCWs located in succession in ascending memory locations

The I/O Concept

Started I/O is Controlled and Processed by the Channel

- CCWs that comprise the CP
  - are mostly contiguous
  - are capable of conditional branching
  - contain flags that instruct CP to (among other things) continue to the next CCW and execute it or to stop
- When a CP stops an appropriate I/O Interrupt is sent to the CPU by the channel

The I/O Concept

Interrupt(s) Signaling Completion are processed by I/O-FLIH

- I/O - FLIH performs the same 'good stuff' as all FLIHs
- The I/O - FLIH chooses further actions based on
  - the stored CSW (X'40')
  - the UCB for the device involved
  - the RQE for this specific operation
  - the I/O Old PSW

Programs Execute CPU Instructions

Sequential and Partitioned Access Methods

- The 'SPAM' assignment
  - usually carried out entirely in MVS/zOS
  - uses Queued and Basic AMs
  - implements Buffering and Blocking concepts

For I/O: New Control Blocks (CB)

- A Unit Control Block (UCB) is provided in the processor's Fixed Storage for EACH I/O device
- One Request Queue Element (RQE) residing in a dynamic storage block is connected to the UCB for each I/O request initiated to that specific device
- This in the case in both a uni-programming and a multi-programming environment

For I/O: New Control Blocks (CB)

- Use the 'UCB' DSECT to identify fields of the UCB

```
UCB      DSECT
UCBIDENT DS    CL3'UCB',X'FF' DOCUMENTATION
UCBCHAN  DS    X'0000' DEVICE ADDRESS IN HEX
UCBFLA   DS    X'00' = FREE, X'FF' = BUSY
UCBFLB   DS    X'00' RESERVED
UCBUCB   DS    A(0)  ADD. NEXT UCB ON CHAIN OR 0 IF THIS = LAST
UCBNAME  DS    CL4'NAME' DEVICE ADDRESS IN EBCDIC
UCBCURQE DS    A(0)  ADDRESS OF RQE CURRENTLY BEING PROCESSED
UCBWARQE DS    A(0)  ADD. FIRST RQE IN WAIT QUEUE,/0 IF NONE THERE
UCBIOSTR DS    A(0)  ADDRESS OF DEVICE'S I/O START ROUTINE
         DS    A(0)  RESERVED
UCBLNTH  EQU   * - UCBIDENT LENGTH OF UCB  (X'20')
```

For I/O: New Control Blocks (CB)

- Use the 'RQE' DSECT to identify fields of the RQE

```
RQE      DSECT
RQEIDENT DS    CL4'RQE ' DOCUMENTATION
RQEIOB   DS    A(0) ADDRESS OF IOB THAT GENERATED THIS RQE
RQEUCB   DS    A(0)      ADD. OF UCB WHEN IN USE.
RQETCB   DS    A(0)      ADD. OF USERS TCB
RQERB    DS    A(0)      ADD. OF RB IN TCB ISSUING I/O REQUEST
RQENRQE  DS    A(0) ADD NEXT RQE IN UCB WAIT QUE./0 IF THIS=LAST
RQEIOECB DS    A(0)      ECB FOR WAIT/POST OF THIS RQE     V6.10
         DS    A(0)      RESERVED                          V6.10
RQELNTH  EQU   *-RQE     LENGTH OF RQE  (X'20')
```


To Start I/O: New Procedures

- The world of 'Real I/O' in the Assist-V simulator environment has four Unit Record devices

DDNAME	'Real I/O' Device Address	
VIRTRDR1	X'000C'	Card Reader
VIRTRDR2	X'000D'	Card Reader
VIRTPRT1	X'000E'	Printer
VIRTPRT2	X'000F'	Printer

To Start I/O: New Procedures

- Assist-V has located these Unit Record devices on Channel 0
- The simulator can also support DASD devices on Channels 1 & 2

To Start I/O: New Procedures

- To begin 'Real I/O' (in SOS)
  - point R1 to an I/O block with non-zero IOBDEVAD field containing the address of a Unit Record device
  - call SVC 0
- In this case SVC 0 will
  - not use XREAD and XPRNT commands
  - BR R12 to the 1st Start I/O Routine

To Start I/O: New Procedures

- Start I/O Routines
  - RQE Enqueue
  - BEGINIO
  - DO-SIO

To Start I/O: New Procedures

Responsibilities of the RQE Enqueue Routine

1. Obtain an RQE from Dynamic Storage
   - Clear it to X'00' and fill required fields
   - Add it to the end of the RQE Queue attached at the UCBWARQE field of the UCB belonging to the device designated in the IOBDEVAD field of the IOB
2. BR entry/exit SVC 1
3. BR R12 to the BEGINIO Routine

To Start I/O: New Procedures

Responsibilities of the BEGINIO Routine

1. Search the UCB chain for the 1st UCB that meets these criteria
   a. The device is NOT-busy
   b. The UCBWARQE field is not zero

   If no such UCBs are found, exit BEGINIO via R12 to Chap/Dispatcher

To Start I/O: New Procedures

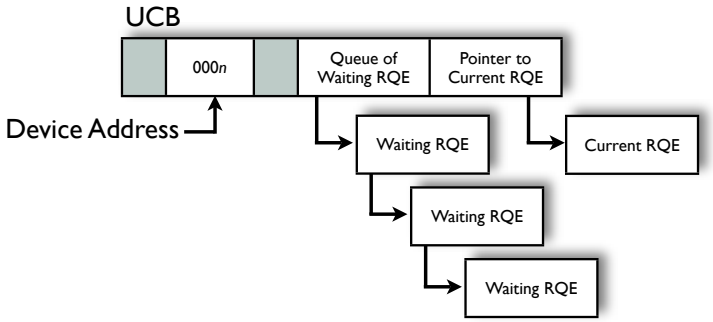
Responsibilities of the BEGINIO Routine

2. Dequeue first RQE from UCBWARQE
3. Point UCBCURQE field to dequeued RQE.
4. BR R12 to this UCB's DOSIO Routine

   The DOSIO Routine will perform the set up necessary to issue an SIO

To Start I/O: New Procedures

Unit Control Block



To Start I/O: New Procedures

Responsibilities of the DOSIO Routine

1. Requirements of DOSIO
   a. A DOSIO routine for each device (UCB)
   b. Each DOSIO requires an SIO and CP
   c. Readers require a 1-CCW CP to read 80 byte input records
   d. Printers require a 2-CCW CP to handle lines of varying length and to control line spacing

To Start I/O: New Procedures

Responsibilities of the DOSIO Routine

A	B	C	res	D
02	A(Input Buffer)	00	00	0050

1	3	1	1	2
Mach. CC	Ignored	40	00	0001
01	A(Print Buffer+1)	00	00	Buf length - 1

(A) CCW Op Code

(B) A(Buffer)

(C) Flags / Chain Cmd

(D) Length

To Start I/O: New Procedures

Responsibilities of the DOSIO Routine

2. Modify the CCWs based on information from the IOB
3. Store A(CP) into CAW (X'48') and issue SIO on device specified in IOBDEVAD
4. Upon return, Test the CC
   If CC=0, do some housekeeping
   Otherwise, (in SOS) Abend
5. BR R12 into BEGINIO through common entry point

I/O Interrupt Processing - I/O FLIH

- Recall:
  - I/O - FLIH performs same 'good stuff' as all FLIH
  - Then examines the stored CSW
- The examination is the first of multiple steps to determine the state of the just-completed I/O operation

I/O Interrupt Processing - I/O FLIH

The Examination of the CSW

- Test Unit and Channel Status bits (32-39,40-47) for 'Bad' bits
- 'Bad' bits present? Terminate the program
  - 'Bad' bits are ANY enabled bits other than
    Channel End
    Control Unit End
    Device End
    Incorrect Length
    Unt Exception

I/O Interrupt Processing - I/O FLIH

The Examination of the CSW

- There are three possible 'end-generating interrupts'
  - Channel end
  - Control Unit end
  - Device end
- Receiving multiple 'ends' is OK
- DEVICE END is required - it proves completion of the operation

I/O Interrupt Processing - I/O FLIH

Examine the I/O Old PSW

- Get A(device) from I/O Old PSW interrupt code field
- Use device address to find proper UCB & current RQE
- Use the current RQE field, RQEIOECB, with a Post Code of X'7F',C'IOS' (indicating the 'I/O Supervisor')
- BR entry & exit SVC 2 to make the program that requested the I/O dispatchable

I/O Interrupt Processing - I/O FLIH

Examine the I/O Old PSW

- Free the CB memory used for the RQE (& return to stack)
- Following Interrupt Processing in the IO-FLIH, exit to the common entry point of BEGINIO.
- BEGINIO can now start any enqueued I/O requests
- If there are no more I/O requests to start, BEGINIO goes to Chap/Dispatcher

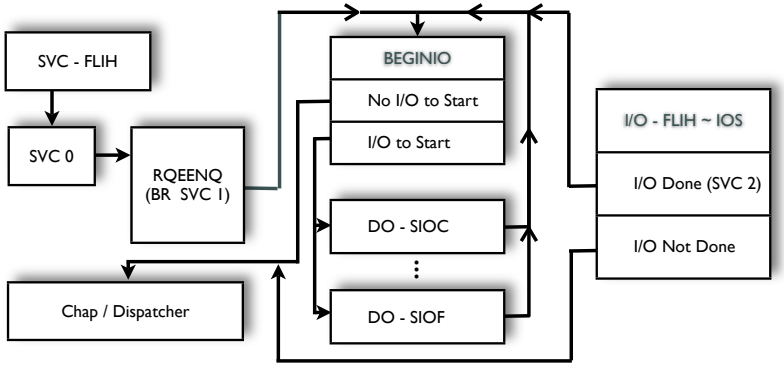
I/O Interrupt Processing - I/O FLIH

A Note on SOS and Chap

- Only some of the SOS assignments have been done with Chap
- It is an SVC that may change priority of TCBs
- Accumulated accounting information may be used to decide where in the TCB chain a particular job (TCB) should be located
- If Chap is active it may change the position (and thereby the priority) of a TCB and then enter the Dispatcher

I/O Interrupt Processing - I/O FLIH

Note: Interrupts which do not present Device End are returned to Chap/Dispatcher immediately rather than passing



Sneak Preview

Highlights

- GPC and OS Remix with
  - IPL
  - SVC
  - Program Interrupts
  - I/O Highlights

Questions

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