SHARE
Technology • Connections • Results

IMS in a 64-bit world.

André Schoeman
BMC Software Inc.

Tuesday, August 3rd, 2010
Session Number 7867

SHARE in Boston

bmcsoftware
BUSINESS RUNS ON I.T.

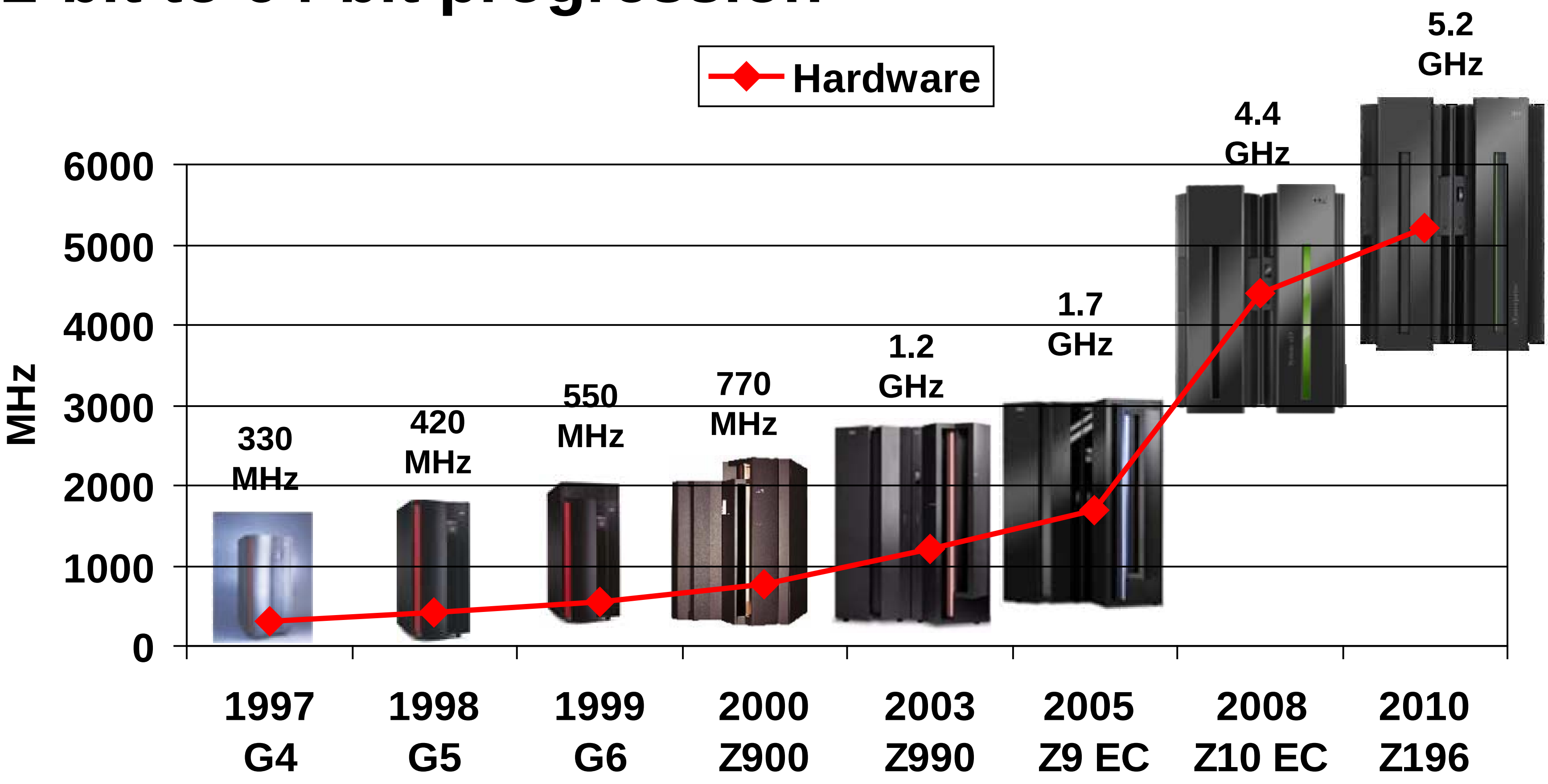
IMS in a 64-bit world: Main topics

- 32-bit to 64-bit progression
  - Hardware
  - Software
    - OS/390 and z/OS
    - IMS
- IMS exploitation
  - Log buffers
  - ACBLIB member cache
  - LSQA relief
  - Fast Path
    - Buffer management
- BMC Software: A vendor/user experience
  - Fast Path/EP Utilities
  - Fast Path/EP Indexer
  - Fast Path/EP Restructure
- “Hints and Tips”
  - Assembler Programming
    - Instructions “structure”
    - Modal instructions
    - Relative instructions
  - Environmental factors
    - Load module addresses
    - Save area formats and chains

32-bit to 64-bit progression

HARDWARE

32-bit to 64-bit progression



- G4 - 1st full-custom CMOS S/390
- Z900 - Full 64-bit z/Architecture
- Z196 - zEnterprise hardware

Some hardware differences

- ESA/390
  - 64-bit PSW
  - 32-bit Control Registers
  - 32-bit GPR's
    - Bits 0 – 31
  - 32-bit Access Registers
  - 4K prefix area (low core)
  - 168 byte LS state entries
  - Some z/Architecture instructions retro fitted to ESA/390

- z/Architecture
  - 128-bit PSW
  - 64-bit Control Registers
  - 64-bit GPR's
    - Bits 0 – 31 = High Order
    - Bits 32 – 63 = Low Order
  - 32-bit Access Registers
  - 8K prefix area (low core)
  - 296 byte LS state entries
  - Lots of new instructions that manipulate 64 bits.

ESA/390 PSW format

0	R	0 0 0	T	I O	E X	KEY	1	M	W	P	A S	C C	PROG MASK	0 0 0 0 0 0 0 0	
0	1	2	5	6	7	8	12	13	14	15	16	18	20	24	31

B A	Instruction address	
32	33	63

z/Architecture PSW format

0	R	0 0 0	T	I O	E X	KEY	0	M	W	P	A S	C C	PROG MASK	0 0 0 0 0 0 0	E X
0	1	2	5	6	7	8	12	13	14	15	16	18	20	24	31

B A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	33																														63

Instruction address	
64	95

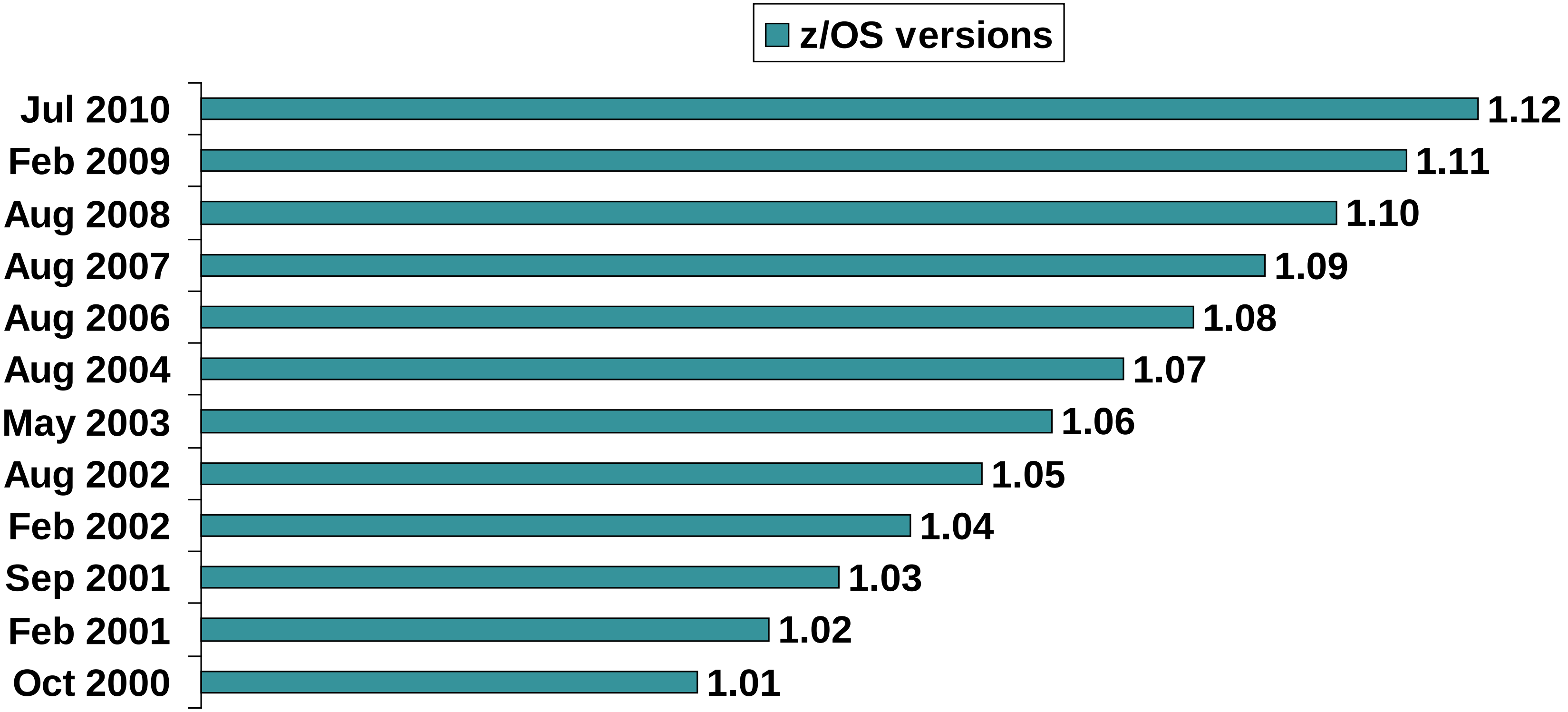
Instruction address (continued)	
96	127

32-bit to 64-bit progression

SOFTWARE

The operating system

32-bit to 64-bit progression



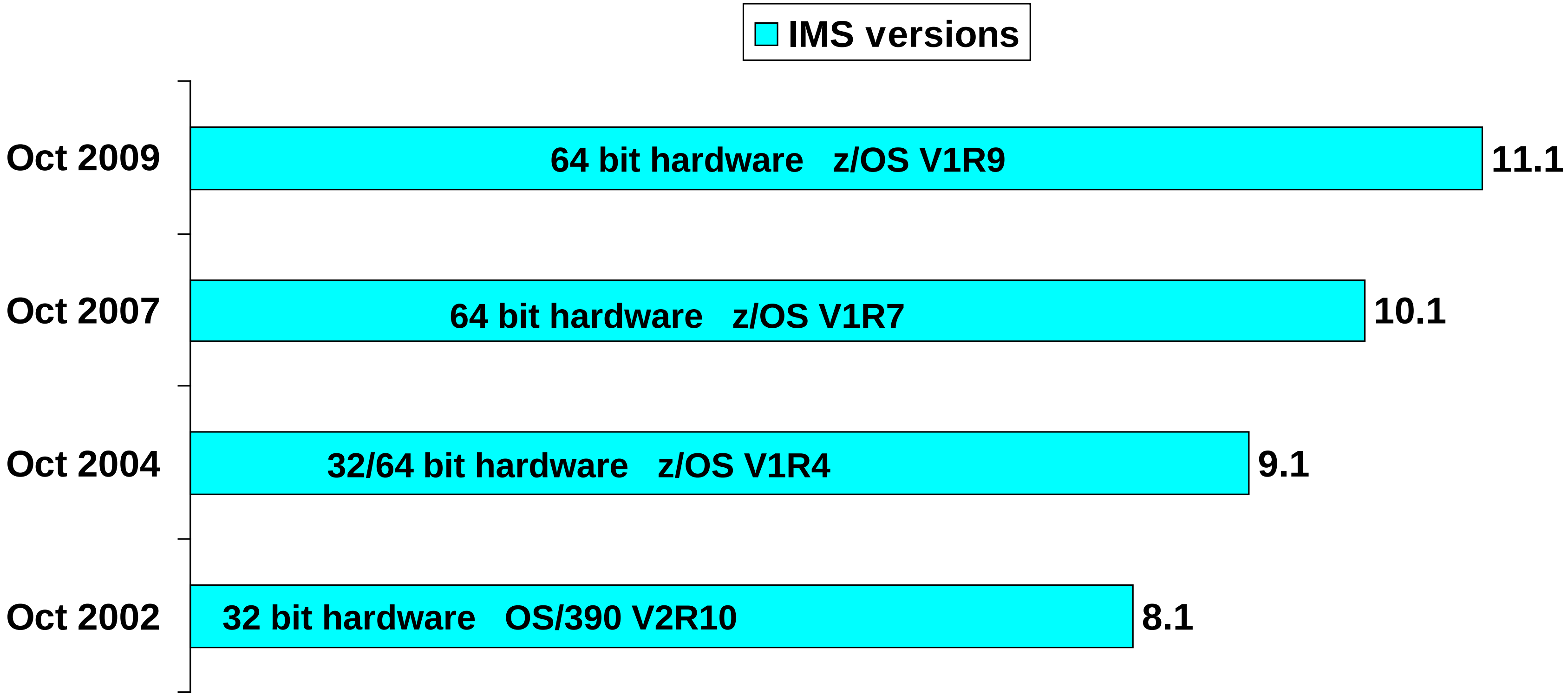
- V1.2 – Initial 64-bit support.
- V1.5 – Shared 64-bit memory obj.
- V1.10 – HCSA (shared COMMON above 2G bar)

32-bit to 64-bit progression

SOFTWARE

The IMS application server

32-bit to 64-bit progression



- V8.1 – Last version to execute on OS/390 V2R10.
- V9.1 – Executes on either 32 bit or 64 bit hardware.
- V10.1 – Requires 64 bit hardware.
- V11.1 – Exploits 64-bit virtual.

IMS exploitation

LOG BUFFERS

IMS exploitation: Log buffers

- Log buffers page fixed in ECSA (31-bit virtual)
  - Performance
- 64-bit real storage backing of 31-bit virtual
  - Introduced in IMS V10
  - Requires z/Architecture (IPL mode = z/Architecture)
  - OLDS block size must be multiple of 4K
  - Environments that are 31-bit real constrained, but have spare 64-bit real capacity, may benefit.
- No 64-bit virtual exploitation (yet).

IMS exploitation

ACBLIB member cache

IMS exploitation: ACBLIB member cache

- ACBLIB members (DMB’s and PSB’s) cached in 64-bit pool
  - Introduced in IMS V11
  - Not all ACBLIB members qualify:
    - Defined as “resident”.
    - DEDB’s
  - Non-resident members are loaded on demand.
  - ACBLIB member caching available in all online environments, but not IMS batch.

IMS exploitation: ACBLIB member cache

- Defined in DFSDFxxx PROCLIB member
  - DATABASE section
  - ACBIN64 parameter specifies pool size in GIG
    - e.g. ACBIN64=8 specifies 64-bit pool 8G in size
    - Dynamic expansion to limit.
    - Cast out on LRU basis.
    - Beneficial to size correctly.
  - Can provide ECSA relief of resident pool
  - CSL not required, except if QUERY POOL command is used.

IMS exploitation: ACBLIB member cache

- At 1st schedule non-resident member loaded into non-resident pool.
  - Also loaded into ACBIN64 pool.

- Next schedule, non-resident member is read from ACBIN64 cache, instead of I/O to ACBLIB.

- ACBIN64 supported by OLC, MOLC and DRD.
  - Does not create/update member in pool.
  - Deletes member from ACBIN64 pool.
  - Next schedule will load new/updated member to ACBIN64 pool.

IMS exploitation: ACBLIB member cache

- QUERY POOL TYPE(ACBIN64) shows pool info
- x’4515’ statistics log record shows ACBIN64 stats (same as in QUERY POOL output)
- New monitor records
  - Type 74, 75, 76 and 77
- New field on region IWAIT report
  - BLR-64BIT

IMS exploitation

LSQA relief

IMS exploitation: LSQA relief

- Some background:
  - IMS performs internal storage management.

  - Tracks module and storage usage via CDE chain(s).
    - CDE blocks z/OS architected in 24-bit LSQA

  - Long CDE chain(s) can exhaust 24-bit storage.
    - S878 abend
      - *z/OS bypasses recovery termination/cleanup routines*
    - S40D IMS termination
      - *Large chunks of orphaned CSA/ECSA*
      - *IPL to fix*
        - *(PC world's **"re-boot to fix"** option not desirable !!)*

IMS exploitation: LSQA relief

- Solution:
  - Introduced in IMS V11
  - New 64-bit (private) Storage Tracking Element (STE)
    - Eliminates 24-bit CDE’s for IPAGES
    - Track IPAGES storage differently.
  - Available to CTRL and DLISAS address spaces
  - APAR PM17966 implements similar relief for some OSAM control block tracking in DLISAS address space.
  - No user specification / activation required.
    - Part of base IMS

IMS exploitation

Fast Path 64-bit Buffer Manager

IMS exploitation: FP 64-bit Buffer Manager

- Introduced in IMS V11
- New buffer manager.
  - Optional to existing buffer manager
  - One or the other is used
- Uses 64-bit storage for DEDB data buffers.
  - MSDB, System, SDEP buffers still in 31-bit ECSA
  - DMHR and other control blocks still in 31-bit ECSA
- Multiple sub-pools within.
  - Different sub-pools for different CI sizes
  - Dynamic contraction / expansion of pool
    - (future IMS release)

IMS exploitation: FP 64-bit Buffer Manager

- To activate:
  - FASTPATH section in DFSDFxxx proclib member
    - FPBP64=Y
    - FPBP64M=<size>
      - *DFS3299I msg if required storage > FPBP64M specification*
  - Synchronize FDBR / XRF parameters, if used.
  - COLD start IMS
    - Must COLD start to switch between buffer managers

- DFS3300I message
  - Shows DBBF, DBFX and BSIZ parameters ignored
    - Used by old buffer manager

IMS exploitation: FP 64-bit Buffer Manager

- Advantages:
  - Multiple sub-pools, 1 for each CI size
    - No unnecessary storage waste.
    - Dynamic management of sub-pools.
    - New CI size without IMS recycle.
  - ECSA relief
    - Old buffer manager has all DEDB buffers in ECSA
  - OBA no longer single treaded
    - Performance
  - Stability
    - Reduced exposure to U1011 and IPL (ECSA fragmentation)

IMS exploitation: FP 64-bit Buffer Manager

- How did it happen ???
  - Dual code path for DL/1 action modules.
    - Duplicated modules with different suffixes, where required
    - Stability
  - Many new modules (all OCO, off course!)
    - DBFDEDB0 has grown in size
      - *ECSA used by larger DBFDEDB0 far less than ECSA freed by 64-bit data buffers.*
  - New anchor points in ESCD
  - Expanded control blocks

IMS exploitation: FP 64-bit Buffer Manager

- QUERY POOL TYPE(FPBP64) shows pool info
  - x’4516’ statistics log record shows same info
- x’4081’ log record contain pool info for WARM or ERE restart.
- x’5945’ log record shows pool statistics
  - Logging not automatic. Must be activated with UPDATE IMS SET(LCLPARM(FPBP64STAT(Y))) command.
  - Default setting = N
  - Not maintained across restart.
- x’5960’ log record has sub-pool management info

BMC Software Inc.

A vendor / user experience with IMS in a 64-bit world.

BMC - Fast Path/EP Utilities

- Supports Fast Path 64-bit Buffer Manager
  - Full offline support
    - Private buffer management
  - Limited online support on z/OS V1R9
    - Analyzer
    - Image Copy
  - Full online support on z/OS V1R10 and higher
    - Online Reorg AND Online Extend
      requires Fast Path/EP installed in CTRL region.
      - *BMC128013I   confirmation message*
    - Private buffer management.
    - Switch to required state for IMS 64-bit data buffer access.

BMC - Fast Path/EP Indexer

- Supports Fast Path 64-bit Buffer Manager
  - Requires z/OS V1R10 or higher
  - Different implementation
    - Affected code (in CTRL region) runs in AMODE(64), regardless of buffer manager
      - *Single code path*
  - Requires version ZPFP39.01.10 (GA Sep 2010) with APAR/PTF BAQ5560/BPQ4932

BMC - Fast Path/EP Restructure

- Available May 2010

- Supports IMS V9 and higher
  - 64-bit hardware and z/Architecture required

- Supports Fast Path 64-bit Buffer Manager
  - Requires 64-bit hardware and z/Architecture mode.
  - Requires z/OS V1R10 (or higher)

- **What does it do ???**
  - Online restructure of DEDB without an outage.
    - Reality check: There is a small outage

BMC - Fast Path/EP Restructure

- Similar in concept to CRF for Full Function databases:
  - Shadow areas.
  - Change Capture hook
  - Focus on structure change(s), not space reclaim.

- Patented technology:
  - Delta Reload for specific areas
  - Change Capture Pause
    - BMCPAUSE

BMC - Fast Path/EP Restructure

- Major features:
  - Four phases
    - PREPARE
    - SHADOW INIT
    - RESTRUCTURE
    - TURNOVER
  - Only affected areas are processed
    - Increased availability

BMC - Fast Path/EP Restructure

- Major features (continued):
  - Only copied UOW’s captured
    - Maximized speed
    - Optimal buffering
  - **Exploits 64-bit technology**
    - **Inter address space CI buffering in 64-bit storage**
    - **Mix of AMODE(31) and AMODE(64)**
    - **Control blocks in 31-bit storage**

BMC - Fast Path/EP Restructure

- Supported structure changes:
    - ADD or REMOVE areas
    - Resizing of areas
    - Randomizer changes
    - Add segments at end of hierarchical path
    - Add SDEPs
    - Add / Change / Remove compression exit.
    - Modify lengths of variable length segments:
        - Decrease minimum length
        - Increase maximum length

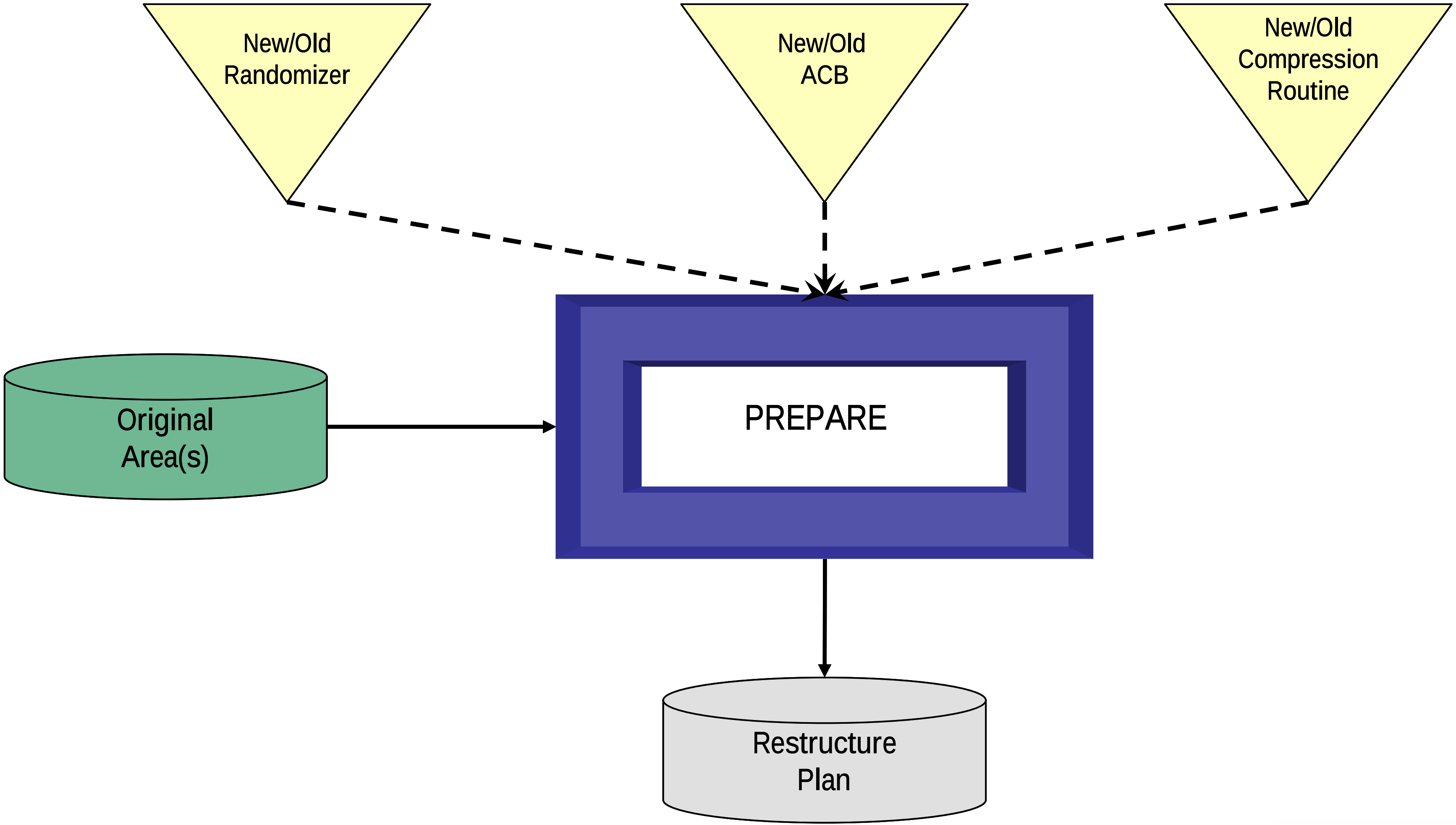
BMC - Fast Path/EP Restructure

- Application affecting restrictions:
  - Cannot add a segment in hierarchical path to existing segment
  - Cannot remove a segment from hierarchical path to existing segment
  - Cannot add a sequence field to existing non-keyed segment
  - Cannot remove a sequence field from existing keyed segment
  - Cannot modify an existing sequence field

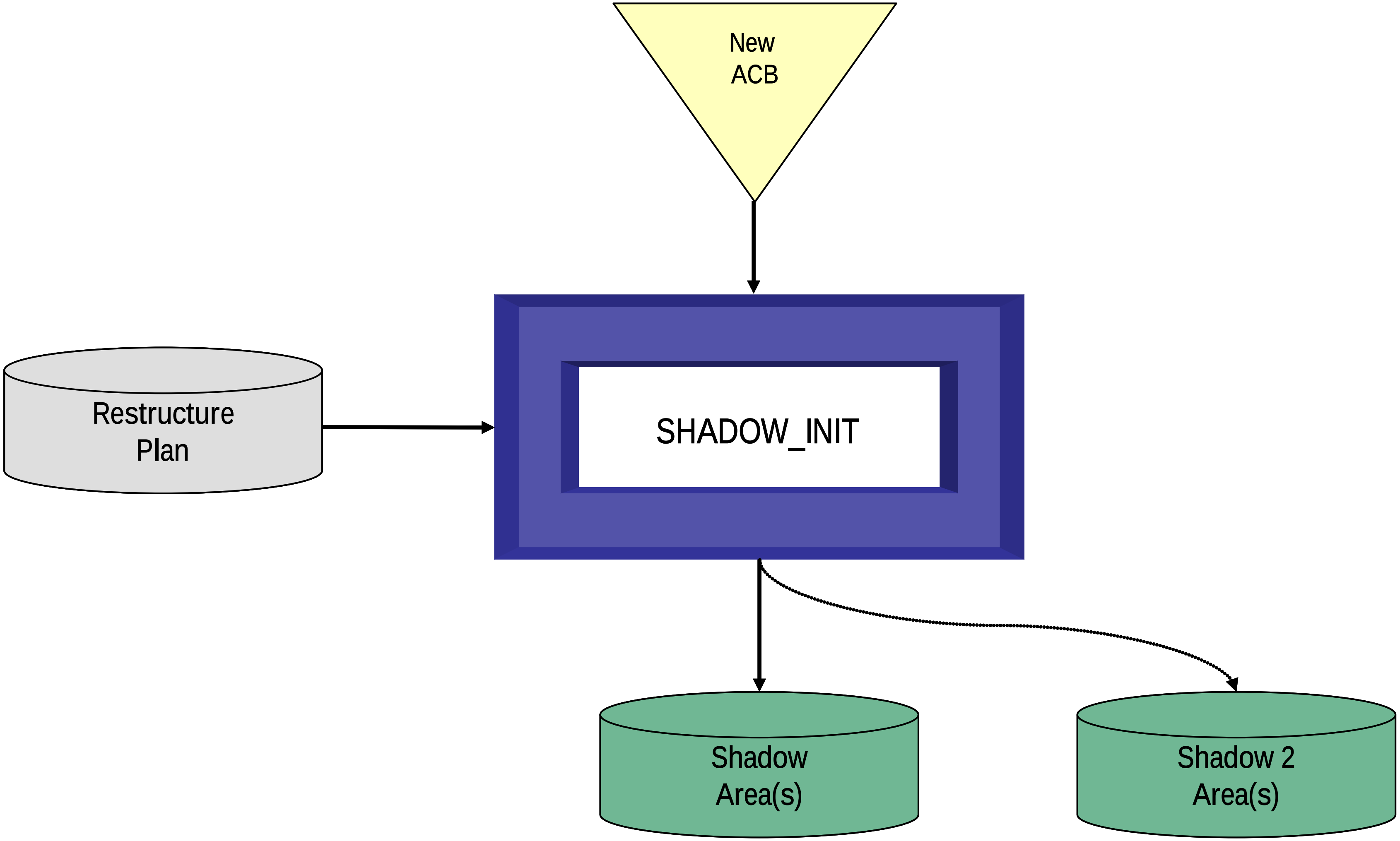
BMC - Fast Path/EP Restructure

- Known operational considerations:
  - DBRC registration mandatory
  - Dual DASD for affected AREAs
    - Triple if secondary shadow selected (image copy option)
  - Each execution supports a single database
  - Physical SDEP placement not maintained
  - SDEP SCAN/DELETE unavailable during process
    - User mod disables function whilst RESTRUCTURE
  - Short outage needed for swap
  - Subset pointer updates not captured on retrieval calls

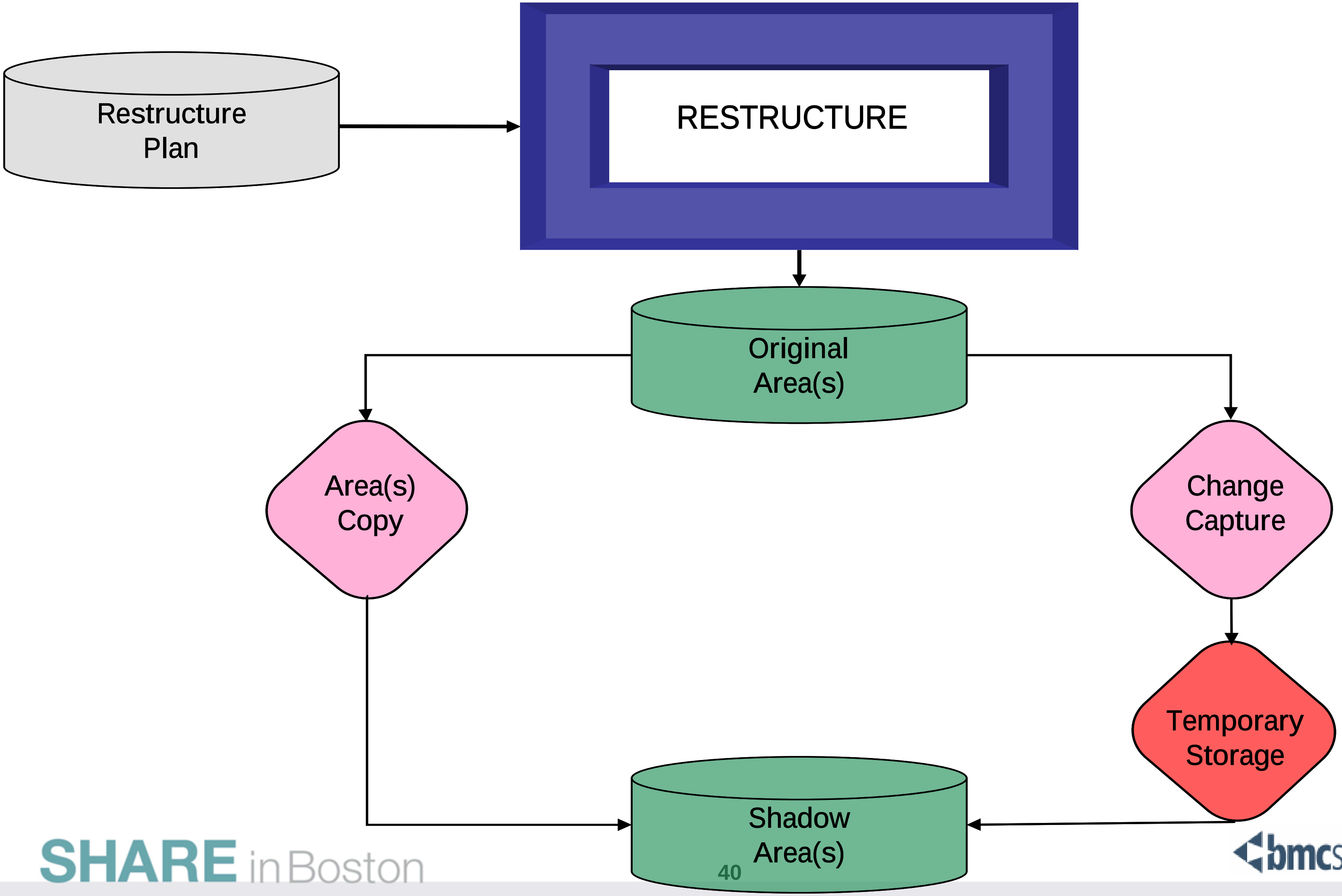
BMC - Fast Path/EP Restructure



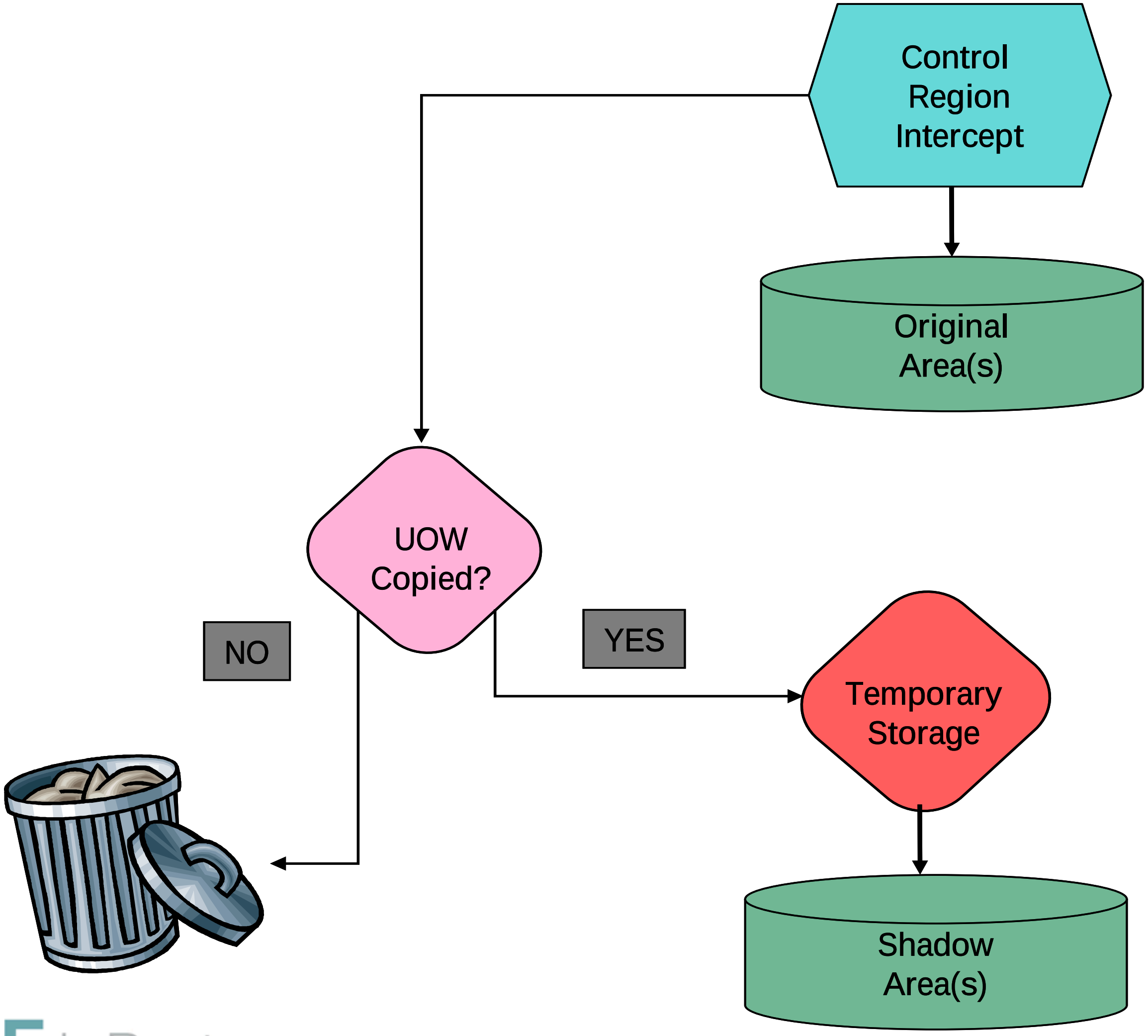
BMC - Fast Path/EP Restructure



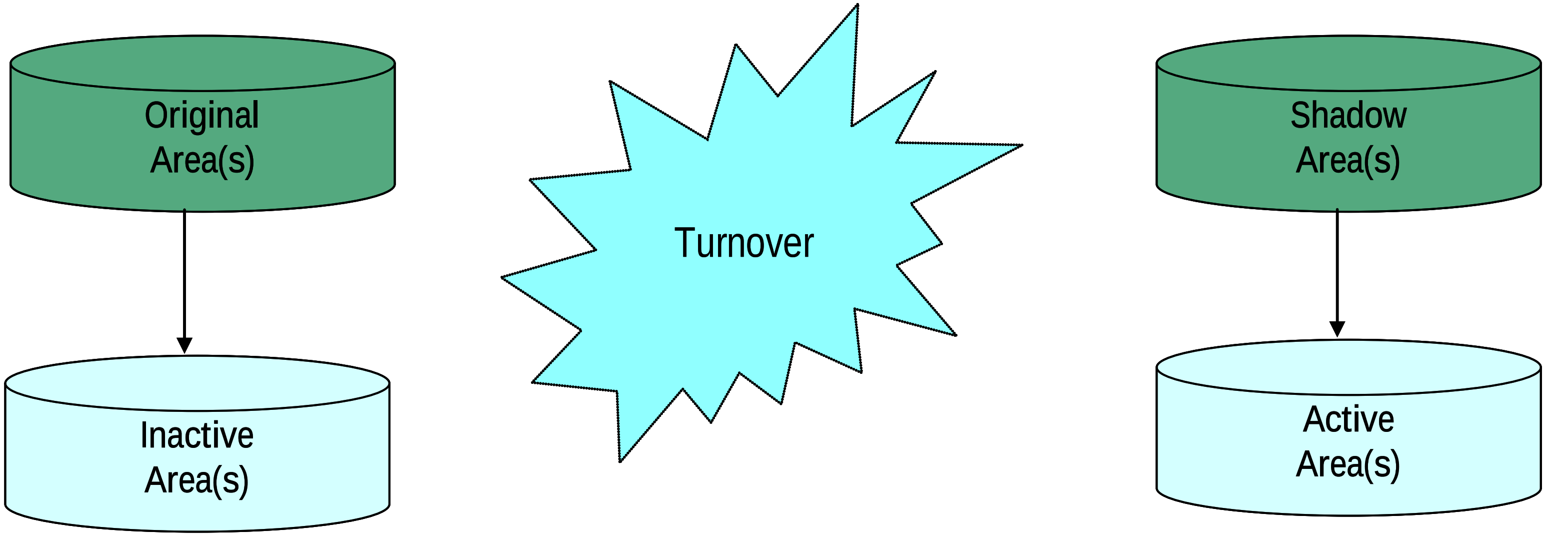
BMC - Fast Path/EP Restructure



BMC - Fast Path/EP Restructure



BMC - Fast Path/EP Restructure



Hints and tips.

Assembler programming

Hints and tips: Assembler Programming

Source instruction “structure”

Hints and tips: Assembler Programming

- Backward compatibility with existing instructions that only manipulate 32 bits in register.

- Instruction code determine bit scope in registers
  - Analogs for 64←64 instructions
    - “G” added to mnemonic
  - Analogs for 64←32 instructions
    - “F” added to mnemonic
  - Existing 32←32 instructions

- PSW bits determine addressing scope
  - Extended Addressing bit
  - Basic Addressing bit

Hints and tips: Assembler Programming

- Instruction analogs easily implemented with macro variables.
  - Conditional assembly of single code set produces different flavour of instructions, depending on the state, or value, of a set of variables.
  - Example:

```
&VG     SETC    ‘’                    /* Init to NULL    */
&VF     SETC    ‘’                    /* Init to NULL    */
                          OR
&VG     SETC    ‘G’                   /* Init to “G”     */
&VF     SETC    ‘F’                   /* Init to “F”     */
                          OR
&VG     SETC    ‘G’                   /* Init to “G”     */
&VF     SETC    ‘’                    /* Init to NULL    */
```

Hints and tips: Assembler Programming

- Generated code:
  - Source code of

```
                A&VG.&VF.R
```

  - will generate

```
                AR
        or
                AGFR
        or
                AGR
```

Hints and tips: Assembler Programming

Modal instructions

Hints and tips: Assembler Programming

- Tri-modal addressing:
  - BASSM, BSM, SAM24, SAM31, SAM64
    - (also TAM to test AMODE status)

- Modal instructions:
  - They behave differently, depending on PSW’s AMODE status.
    - BALR, BASR, BRAS, BRASL, LA, MVCL etc.
      - *See chapter 7 in POPS manual.*

Hints and tips: Assembler Programming

Relative addressing

Hints and tips: Assembler Programming

- Relative addressing: What does it mean ???
    - Different concept: (well, sort of …)
        - “Old way” has fixed base register.
            - *Everything addressed from base register is relative to that base.*
        - Relative instructions use variable base (PSW).
            - *Moving target*
            - *Everything addressed from PSW is relative to the PSW.*
            - *Cannot address “odd” addresses.*
                - *Target must be at least on halfword boundary* **ASMA058E Invalid relative address -** *xxxxxxxx*
            - *Target of absolute value generates warning message*
                - **ASMA056W Absolute value found when relocatable value expected -** *xxxxxxxx*

Hints and tips: Assembler Programming

- Relative addressing (continued):
  - 00000340 C010 0000 01CC   000006D8   2822
    LARL R1,=C'#PFPCCBI'   /* Literal in pool   */
  - 0000010C A7F4 FFF9   000000FE   2424
    J   MAIN0800   /* Go exit ....   */
  - Watch out for existing MVS interface macros.
    - More than often generates code that requires base register, usually for literal pool, often for label reference.
      - *Base register for literal pool good idea.*
    - Its not always what it seems…….

IMS in a 64-bit world

Hints and tips.

Environmental factors

Hints and tips: Environmental factors

Load module addresses

Hints and tips: Environmental Factors

- Load module addresses, as returned by LOAD, has AMODE bits set:
  - Bits 0-31 is zero (if AMODE(64))
  - Bit 32 shows AMODE(24|31)
  - Bit 63 shows AMODE(64)
  - Some modal instructions ignore / process these bits properly.
- Executable code may address data above 2G bar, but CANNOT run there (for now)
  - BASR with target address where bit 32 is on, results in S0C4 abend code 3A or 3B
    - When in AMODE(64) status
  - BASR with target adress where bit 63 is on, results in S0C1 or S0C5

Hints and tips: Environmental Factors

- R0 contents after successful LOAD of load module with AMODE(24) attribute
  - Bits 00 to 31 unpredictable
  - Bits 32 to 39 = zero
  - Bits 40 to 63 = Entry Point Address (24 bit)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
00																															31

0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
32							39	40																							63

Hints and tips: Environmental Factors

- R0 contents after successful LOAD of load module with AMODE(31) attribute
  - Bits 00 to 31 unpredictable
  - Bit 32 = 1
  - Bits 33 to 63 = Entry Point Address (31 bit)

?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
00																															31

1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
32	33																														63

Hints and tips: Environmental Factors

- R0 contents after successful LOAD of load module with AMODE(64) attribute
  - Bits 00 to 31 always zero
  - Bit 32 = 0
  - Bits 33 to 63 = Entry Point Address (31 bit)
  - Bit 63 = 1

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
00																															31

0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1
32	33																														63

Hints and tips: Environmental Factors

- z/OS V1R11 (and later) allows LOAD into 64-bit virtual storage
  - Directed load only
    - z/OS does not track it
    - User must acquire storage 1st, then request LOAD to load the module at that location.
  - Non executable code only (logically)
    - Still cannot execute above 2G bar

Hints and tips: Environmental factors

Save area formats

and

chains.

Hints and tips: Environmental Factors

- Various save area formats
  - SYS1.MACLIB(IHASAVER) provides mapping
- F1SA format:
  - 18 fullwords (72 bytes) in length
  - Stores 32-bit values (GPR’s only)
  - Utilizes 32-bit back / forward pointers.
- F4SA format:
  - 36 fullwords (144 bytes) in length
  - Stores 64-bit values (GPR’s only ??)
  - Utilizes 64-bit back / forward pointers.

Hints and tips: Environmental Factors

- F5SA format:
  - 54 fullwords (216 bytes) in length
  - Stores 64-bit values (GPR’s only)
    - Additional room for 32-bit values (high order)
  - Utilizes 64-bit back / forward pointers.

- F7SA format:
  - 54 fullwords (216 bytes) in length
  - Stores 64-bit values (GPR’s)
  - Stores 32-bit values (AR’s)
  - Utilizes 64-bit back / forward pointers.

Hints and tips: Environmental Factors

- Linkage stack:
  - Associated with dispatchable unit (read TCB)
  - No need 2 know the format
  - BAKR, PR, PC instructions

Hints and tips: Environmental Factors

- Save area chains
  - Sequential, fixed format

  - Sequential (or not), variable format
    - Facilitates R13 mapping of working storage

  - Careful when hooking different type into existing chain !!

  - IMS has pre-allocated save area chains (sequential, fixed format)
    - Uses 2 F1SA areas, where required
      - *Might see (what appears to be) every second set empty in chain*
      - *Reduces cascading call depth*

  - Don’t try and build F1SA chain in 64-bit storage !!

Happy coding !!!

Questions ???

Andre_Schoeman@BMC.COM