PowerPC 74xx Architecture 32-Bit Addressing Modes

Porting Plan 9 to the PowerPC 74xx Architecture

Adam Wolbach
awolbach@andrew.cmu.edu
15-412 Operating Systems Practicum

Abbreviations

Memory

EA	Effective Address	(32-bit)
VA	Virtual Address	(52-bit)
RA	Real Address	(32-bit)
MSR	Machine State Register	
SDR1	Storage Description Register 1	

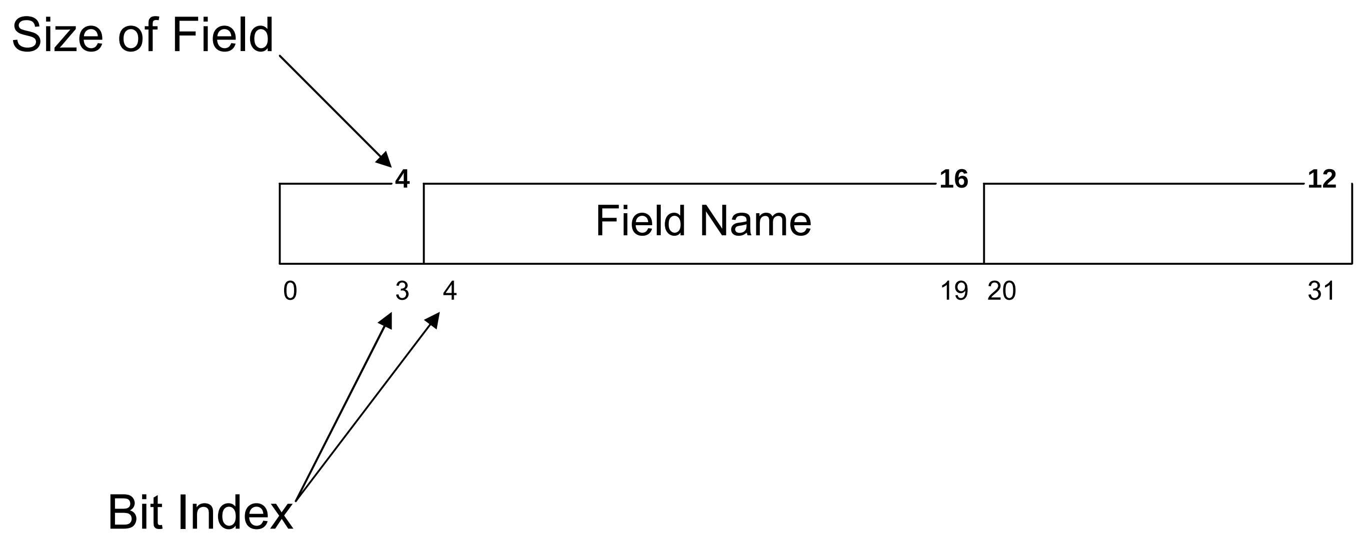
Base Mathematics

0xFFFF	FFFF in Base 16/Hexadecimal
0b1111	1111 in Base 2/Binary

Arithmetic

X || Y	Concatenate X with Y
X & Y	X (bitwise AND) Y
X | Y	X (bitwise OR) Y
X ^ Y	X (bitwise eXclusive OR) Y
~X	bitwise NOT X (complement)
^{Y}X	Repeat bit X, Y times (e.g., $^{3}0$ = 000)

Register Abbreviations



ABC_{XX} Denotes XX bit of register ABC

Addressing Overview

- Three primary mechanisms
  - Real Addressing Mode
  - Block Address Translation (BAT)
  - Segmented Address Translation (SAT)
    - Ordinary Segment Translation
    - Direct-Store Segment Translation
- MSR_{IR} value controls instruction fetches
- MSR_{DR} value controls data accesses

Machine State Register (32-Bit)

SF	///	POW	/	ILE	EE	PR	FP	ME	FE0	SE	BE	FE1	/	IP	IR	DR	//	RI	LE
0	1 12	13		15	16	17	18	19	20	21	22	23	24	25	26	27	28 29	30	31

/ = Reserved

- Controls many important system flags
  - EE[16]: External Enable (Interrupts)
    - If set, external interruption allowed (e.g. Keyboard, “Timer”)
  - PR[17]: Problem State (User Mode)
    - If set, processor can only execute non-privileged instructions
  - IR[26]/DR[27]: Instruction Relocate/Data Relocate
    - If set, Instruction/Data address translation mechanisms on
  - RI[28]: Recoverable Interrupt
    - If set, a resume to regular execution possible

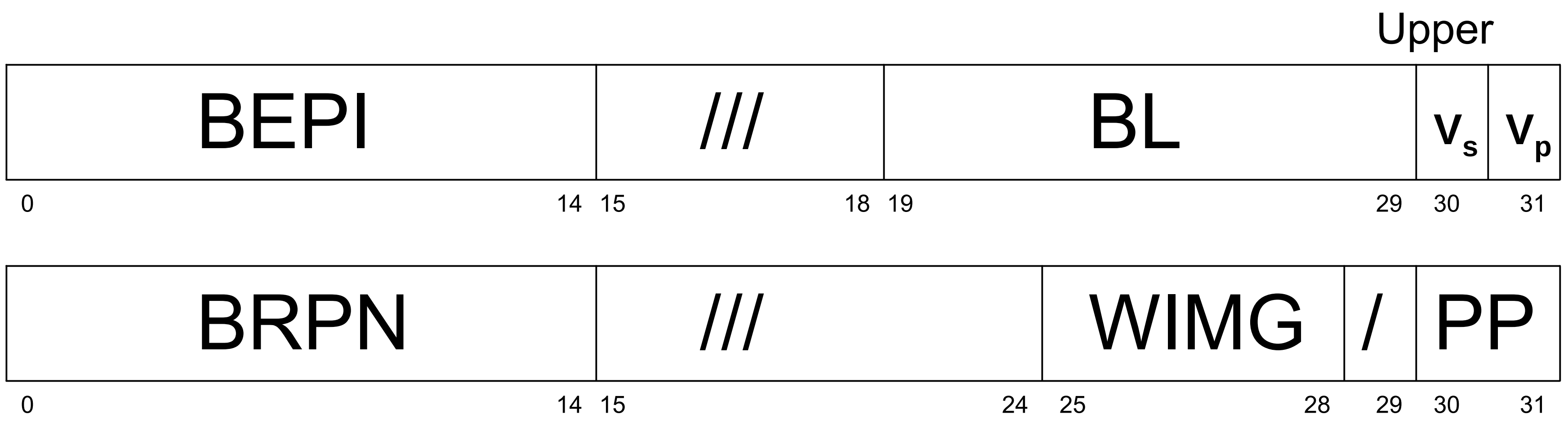
Real Addressing Mode

- EA == RA to the processor
  - Bypasses all storage protection checks/translation
- MSR_{IR} = 0 results in real addressing mode for instruction fetches (only type of access)
- MSR_{DR} = 0 results in real addressing mode for any data accesses, read or write
- MSR_{IR} and MSR_{DR} can exist in any combination of settings

Block Address Translation

- Method of directly mapping large virtual address spaces to contiguous real memory addresses
  - Length must be a power of 2, from 2^{17} to 2^{28}
    - Controlled by a mask field in the upper register
    - Block Length = $2^{17 + (\text{\# of bits in mask set})}$
  - Alignment must occur on a multiple of its length
- Defined by 8 CPU special-purpose register pairs
  - 4 IBAT (Instruction), 4 DBAT (Data)
  - Each pair consists of upper and lower register
- Enabled if MSR_{IR} and/or $MSR_{DR} = 1$
- Great for memory-mapping
  - Display buffer, kernel memory, etc.

BAT Register Pair



BEPI	Block Effective Page Index
BL	Block-length Mask
	e.g., 0x003 = $2^{17+2=19}$ (512 KB)
BRPN	Block Real Page Number
PP	Protection bits for BAT area
	00 = No Access, x1 = Read Only, 10 = Read/Write
V_s	Supervisor state valid bit -- allows root access
V_p	Problem state valid bit -- allows user access
WIMG	Storage Access Controls

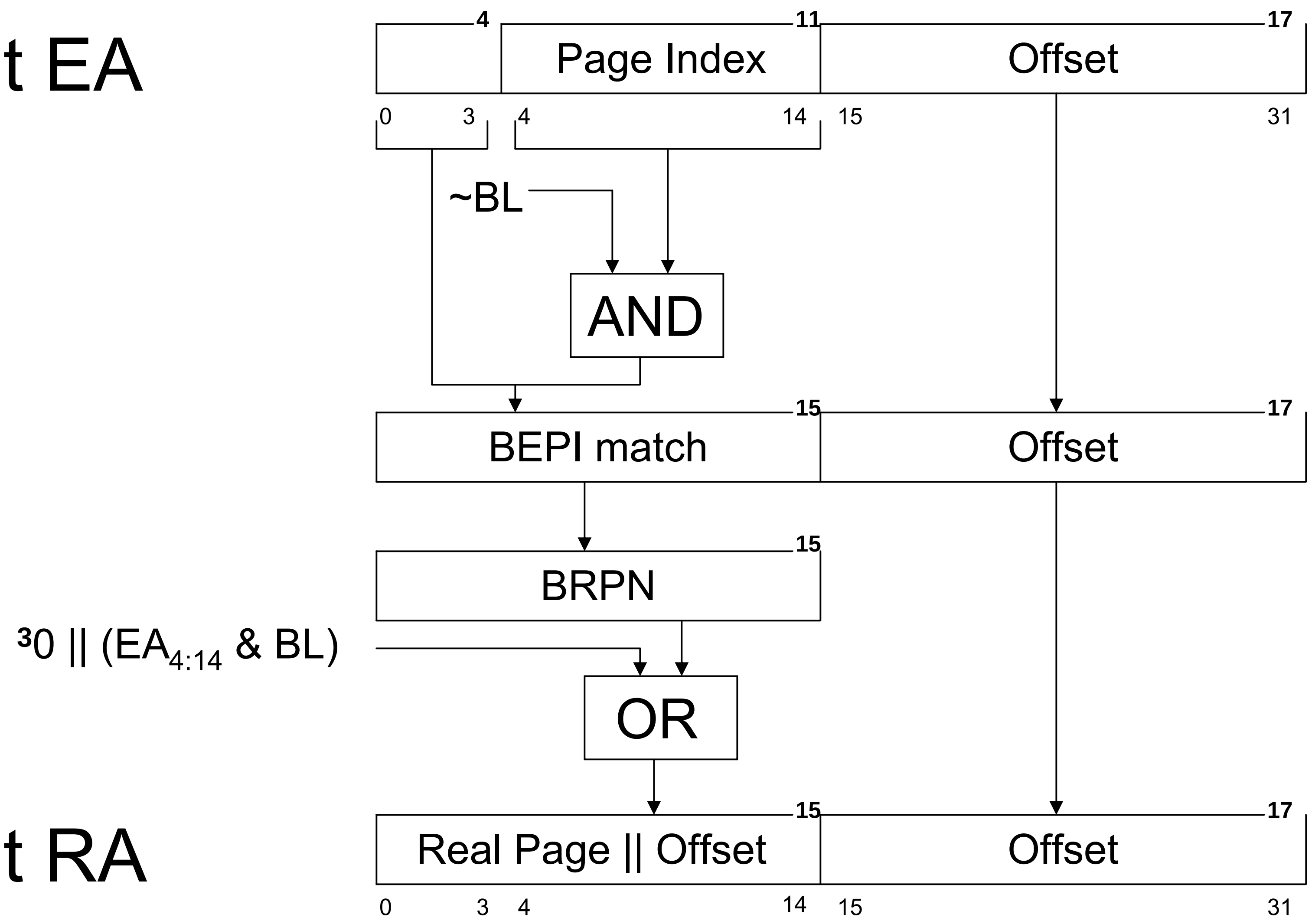
BAT Register Validation

- BAT register valid if these conditions hold:
  - $MSR_{IR} \mid MSR_{DR} = 1$
  - $(V_s \,\&\, \sim MSR_{PR}) \mid (V_p \,\&\, MSR_{PR}) = 1$
  - Cannot overlap any other register’s EA range
    - Unless they cannot be valid at the same time, as per the relation above
    - Translation effects undefined, and probably horrendous, if conflicting memory state exists
- Page Fault Interrupt on PP R/W permissions fail

BAT Translation Method

32-bit EA

32-bit RA



BAT Lookup

- Registers not indexed by bits, but rather searched sequentially by access type
- Address match (EA covered by BAT) if:
  - $EA_{0:3}$ || ($EA_{4:14}$ & ~BL) = BEPI
    - 15 bits [0-14] needed at most to determine block starting address because minimum BAT size is 2^{17}
    - 4 highest order bits not needed in masking because blocks cannot be this large
- BRPN then OR'd with [$^{3}0$ || ($EA_{4:14}$ & BL)] to get remaining page bits from EA
- Offset ($EA_{15:31}$) added, untouched

Example – Data Access

Segmented Address Translation

- Storage divided into 256 MB (2^{28}) segments, of ordinary or direct-store type
  - Ordinary segments controlled by setting of relocate bits MSR_{IR} and MSR_{DR}
    - Used as storage protection
  - Direct-store segments used for access to I/O
    - EA sent to device with key check modification
    - MSR_{DR} must be set
- Segments defined by 16 register “table”

Segment Register (Ordinary)

T	K_s	K_p	///	VSID
0	1	2		8 … 31

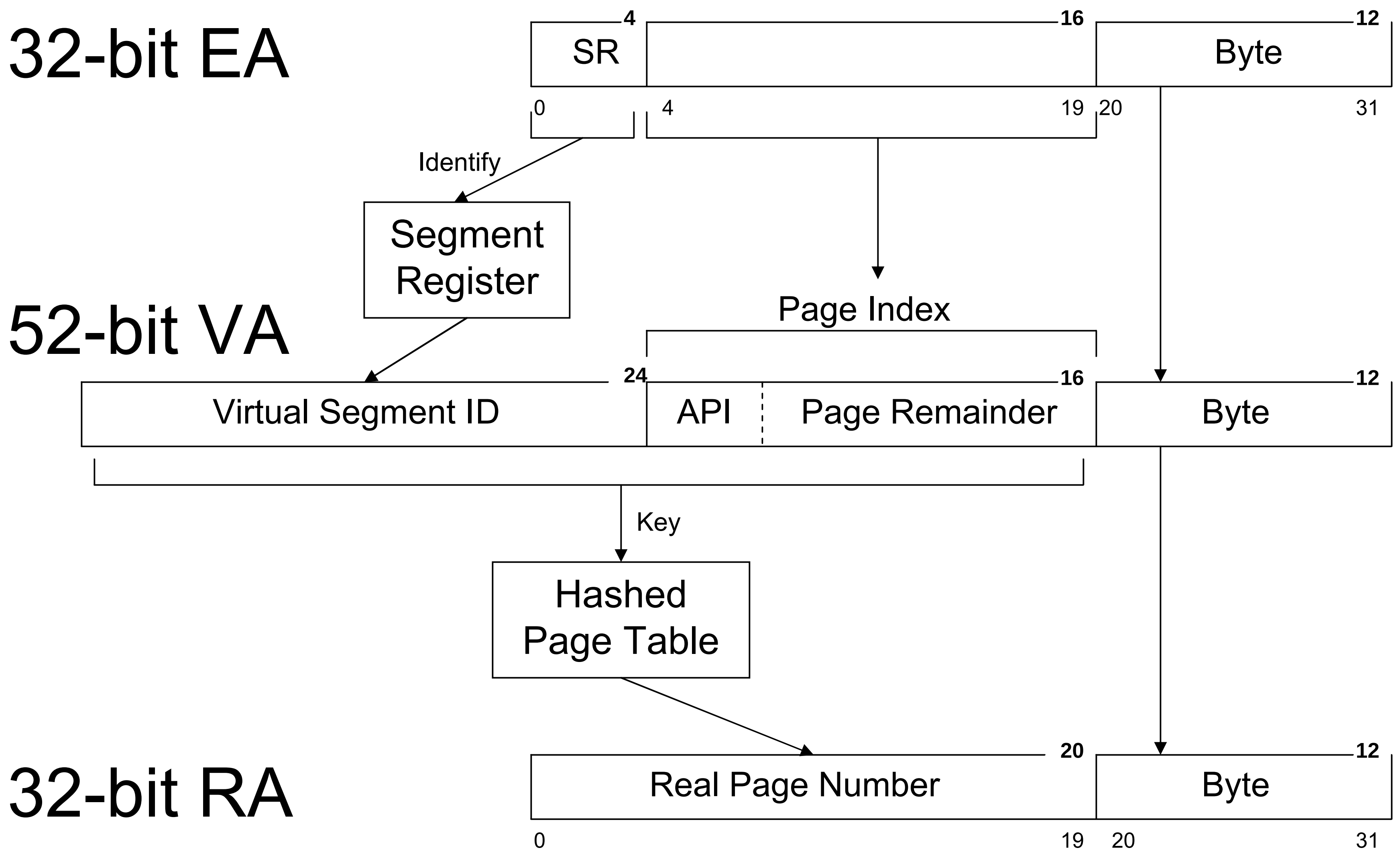
T	= 0, Direct Store off
K_s	Supervisor state storage key (allows supervisor access)
K_p	Problem state storage key (allows user access)
VSID	Virtual Segment ID (24-bit)

Segment Register (Direct-Store)

T	K_s	K_p	BUID	*controller specific*
0	1	2	3	12 … 31

T	= 1, Direct Store on
K_s	Supervisor state storage key
K_p	Problem state storage key
BUID	Bus Unit ID
cs	Device dependent data for I/O

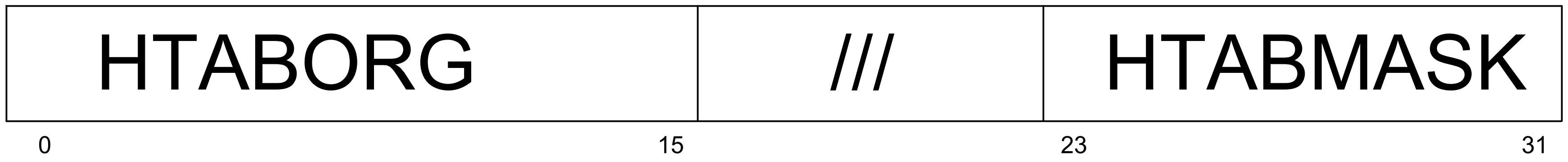
Segment EA to RA Translation



Hashed Page Table

- Variable-sized data structure that hashes between virtual page numbers and real page numbers
  - Must be aligned on its 2^n size, where $16 \le n \le 25$
- Contains 2^{n-6} 64-byte Page Table Entry Groups
  - Each PTEG has 8 PTE entries, each 8 bytes long
  - Important to balance: Size of PT and Page Fault Rate
- Exists in main memory
  - RA and size defined by Storage Description Register 1
  - n, and thus the number of PTEG's, controlled by OS
- Architecture neutral as to # of PT's allowed

Storage Description Register 1 (32-Bit)



HTABORG[0-15]	Real Address of Page Table (Aligned on 2^{16} byte boundary, meaning minimum size is 64KB)
HTABMASK[23-31]	Mask for Page Table Address (e.g., 0x007 strips 3 bits off of the hash to allow for 2^{10+3} PTEGs)

Hashing VA's to RA's

- Key indexed by (VSID derived from segment register || EA Page Index)
  - 40-bit key hashes to 20-bit Real Page Number
  - High-order 6 bits of EA Page Index referred to as Abbreviated Page Index, stored in PTE
    - API resolves issues with hash function using less than all 16 bits of the page index by comparing the PTE's API with the EA's API, which are the bits potentially not used in the hash
- If the primary hashing of the key fails, a secondary hash is attempted using the complement of the original key as its key
- If that fails, a Page Fault Interrupt is taken

Page Table Entry

V	VSID	H	API
0	1 ... 24	25	26 ... 31

RPN	///	R	C	WIMG	/	PP
0 ... 19		23	24	25 ... 28	29	30 ... 31

API	Abbreviated Page Index	*(PTE Collision Disambiguation)*
C	Change Bit	
H	Primary / Secondary Hash	
PP	Page Protection Bits	
	(00 = No Access, x1 = Read Only, 10 = Read/Write)	
R	Reference Bit	
RPN	Real Page Number	
V	Valid Bit	
VSID	Virtual Segment ID	*(PTE Collision Disambiguation)*
WIMG	Storage Access Control	*(Cache Control)*

Hashing VA (Primary)

- 1) Perform following computation on parameters: $VSID_{5:23}$ ^ ($^{3}0$ || $EA_{4:19}$)
  - Denote this as N
  - Note that $EA_{4:19}$ = 16-bit Page Index
- 2) Create following address through concatenations:
  - $SDR1_{0:6}$ || [($N_{0:8}$ & $SDR1_{23:31}$) | $SDR1_{7:15}$] || $N_{9:18}$ || $^{6}0$
  - Note that, at minimum, 10 lower-order bits of N/Page Index identify a unique PTEG
- 3) This identifies a PTEG. Test PTE's inside of it for:
  - $PTE_H = 0$
  - $PTE_v = 1$
  - $PTE_{VSID} = VA_{0:23}$
  - $PTE_{API} = VA_{24:29}$
- 4) If PTE found build Real Address, else proceed to Secondary Hash

Hashing VA (Secondary)

- 1) Perform following computation on parameters: $\sim(VSID_{5:23} \wedge ({}^{3}0 \,||\, EA_{4:19}))$
  - Denote this as N
  - Note that $EA_{4:19}$ = 16-bit Page Index
- 2) Create following address through concatenations:
  - $SDR1_{0:6} \,||\, [\, (N_{0:8} \,\&\, SDR1_{23:31}) \,|\, SDR1_{7:15} \,] \,||\, N_{9:18} \,||\, {}^{6}0$
  - Note that, at minimum, 10 lower-order bits of N/Page Index identify a unique PTEG
- 3) This identifies a PTEG. Test PTE's inside of it for:
  - $PTE_H = 1$
  - $PTE_v = 1$
  - $PTE_{VSID} = VA_{0:23}$
  - $PTE_{API} = VA_{24:29}$
- 4) If PTE found build Real Address, else proceed to Secondary Hash
- 5) Else, a Page Fault Interrupt is issued, OS must deal

Forming RA

- If the Page Table search succeeds, the RA is formed by concatenating the RPN from the PTE with bits 20:31 of the Effective Address (the “Byte”/offset)
- Failure results in Page Fault Interrupt of the access type
  - Instruction Storage Interrupt
  - Data Storage Interrupt

Example – Data Access

A Note on Storage Control

- WIMG bits in BAT registers / PTE’s
  - W – Write-through
    - Stores updates to cache to home storage location
  - I – Caching Inhibited
    - Ignores on-board caches
  - M – Memory Coherence
    - Forces hardware data coherence, allowing improved performance in systems in which accesses to storage kept consistent by hardware are slower than accesses to storage not kept consistent, assuming software can enforce the required consistency. If set, hardware must enforce data coherence.
    - Paraphrased from The PowerPC Architecture
  - G – Guarded Memory
    - If set, prevents speculative execution (prefetching)
    - Not applicable to Instruction BAT entries

Which does the processor use?

- Segment Registers and BAT Registers accessed in parallel, with BAT taking precedence if both translations found valid
- If neither lookup is found to be valid, a Page Fault Interrupt is generated and the OS must deal with the problem

Sources

- The PowerPC Architecture: A Specification For a New Family of RISC Processors, Morgan Kaufmann Publishers, San Francisco, 1994