Continue

Structured computer organization solutions manual pdf

1 SOLUTIONS MANUAL COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION WILLIAM STALLINGS 2 TABLE OF CONTENTS Chapter 1 Introduction...5 Chapter 2 Computer Evolution and Performance...6 Chapter 3 Computer Function and Interconnection...14 Chapter 4 Cache Memory...19 Chapter 5 Internal Memory...32 Chapter 6 External Memory...38 Chapter 7 Input/Output...43 Chapter 8 Operating System Support...50 Chapter 9 Computer Arithmetic...57 Chapter 10 Instruction Sets: Characteristics and Functions...69 Chapter 11 Instruction Sets: Addressing Modes and Formats...80 Chapter 12 Processor Structure and Function...85 Chapter 13 Reduced Instruction Set Computers...92 Chapter 14 Instruction-Level Parallelism and Superscalar Processors...97 Chapter 15 Control Unit Operation Chapter 16 Microprogrammed Control Chapter 17 Parallel Processing Chapter 18 Multicore Computers Chapter 19 Number Systems Chapter 20 Digital Logic Chapter 21 The IA-64 Architecture Appendix B Assembly Language and Related Topics 3 CHAPTER 1 INTRODUCTION A NSWERS TO Q UESTIONS 1.1 Computer architecture refers to those attributes of a system visible to a programmer or, put another way, those attributes that have a direct impact on the logical execution of a program. Computer organization refers to the operational units and their interconnections that realize the architectural specifications. Examples of architectural attributes include the instruction set, the number of bits used to represent various data types (e.g., numbers, characters), I/O mechanisms, and techniques for addressing memory. Organizational attributes include those hardware details transparent to the programmer, such as control signals; interfaces between the computer and peripherals; and the memory technology used. 1.2 Computer structure refers to the way in which the components of a computer are interrelated. Computer function refers to the operation of each individual component as part of the structure. 1.3 Data processing; data storage; data movement; and control. 1.4 Central processing unit (CPU): Controls the operation of the computer and performs its data processing functions; often simply referred to as processor. Main memory: Stores data. I/O: Moves data between the computer and its external environment. System interconnection: Some mechanism that provides for communication among CPU, main memory, and I/O. A common example of system interconnection is by means of a system bus, consisting of a number of conducting wires to which all the other components attach. 1.5 Control unit: Controls the operation of the CPU and hence the computer Arithmetic and logic unit (ALU): Performs the computer s data processing functions Registers: Provides storage internal to the CPU CPU interconnection: Some mechanism that provides for communication among the control unit, ALU, and registers -5- 4 CHAPTER 2 COMPUTER EVOLUTION AND PERFORMANCE A NSWERS TO Q UESTIONS 2.1 In a stored program computer, programs are represented in a form suitable for storing in memory alongside the data. The computer gets its instructions by reading them from memory, and a program can be set or altered by setting the values of a portion of memory. 2.2 A main memory, which stores both data and instructions: an arithmetic and logic unit (ALU) capable of operating on binary data; a control unit, which interprets the instructions in memory and causes them to be executed; and input and output (I/O) equipment operated by the control unit. 2.3 Gates, memory cells, and interconnections among gates and memory cells. 2.4 Moore observed that the number of transistors that could be put on a single chip was doubling every year and correctly predicted that this pace would continue into the near future. 2.5 Similar or identical instruction set: In many cases, the same set of machine instructions is supported on all members of the family. Thus, a program that executes on one machine will also execute on any other. Similar or identical operating system: The same basic operating system is available for all family members. Increasing speed: The rate of instruction execution increases in going from lower to higher family members. Increasing number of I/O ports: In going from lower to higher family members. Increasing memory size: In going from lower to higher family members. Increasing cost: In going from lower to higher family members. 2.6 In a microprocessor, all of the components of the CPU are on a single chip. A NSWERS TO P ROBLEMS 2.1 This program is developed in [HAYE98]. The vectors A, B, and C are each stored in 1,000 contiguous locations in memory, beginning at locations 1001, 2001, and 3001, respectively. The program begins with the left half of location 3. A counting variable N is set to 999 and decremented after each step until it reaches 1. Thus, the vectors are processed from high location to low location. -6- 5 Location Instruction Comments Constant (Count N) 1 1 Constant Constant 3L LOAD M(2000) Transfer A(I) to AC 3R ADD M(3000) Compute A(I) + B(I) 4L STOR M(4000) Transfer sum to C(I) 4R LOAD M(0) Load count N 5L SUB M(1) Decrement N by 1 5R JUMP+ M(6, 20:39) Test N and branch to 6R if nonnegative 6L JUMP M(6, 0:19) Halt 6R STOR M(3, 8:19) Update N 7L ADD M(1) Increment AC by 1 7R ADD M(2) 8L STOR M(3, 8:19) Modify address in 3L 8R ADD M(2) 9L STOR M(4, 8:19) Modify address in 4L 10L JUMP M(3, 0:19) Branch to 3L 2.2 a. Opcode Operand b. First, the CPU must make access memory to fetch the instruction. The instruction contains the address of the data we want to load. During the execute phase accesses memory to load the data value located at that address for a total of two trips to memory. 2.3 To read a value from memory, the CPU puts the address of the value it wants into the MAR. The CPU then asserts the Read control line to memory and places the address on the address bus. Memory places the contents of the memory location passed on the data bus. This data is then transferred to the MBR. To write a value to memory, the CPU puts the address of the value it wants to write into the MAR. The CPU also places the data it wants to write into the MBR. The CPU then asserts the Write control line to memory and places the address on the address bus and the data on the data bus. Memory transfers the data on the data bus into the corresponding memory location. -7- 6 2.4 Address 08A 08B 08C 08D Contents LOAD M(0FA) STOR M(0FB) LOAD M(0FA) JUMP +M(08D) LOAD M(0FA) STOR M(0FB) This program will store the absolute value of content at memory location 0FA into memory location 0FB. 2.5 All data paths to/from MBR are 40 bits. All data paths to/from MAR are 12 bits. Paths to/from AC are 40 bits. Paths to/from MQ are 40 bits. 2.6 The purpose is to increase performance. When an address is presented to a memory module, there is some time delay before the read or write operation can be performed. While this is happening, an address can be presented to the other module. For a series of requests for successive words, the maximum rate is doubled. 2.7 The discrepancy can be explained by noting that other system components aside from clock speed make a big difference in overall system speed. In particular, memory systems and advances in I/O processing contribute to the performance ratio. A system is only as fast as its slowest link. In recent years, the bottlenecks have been the performance of memory modules and bus speed. 2.8 As noted in the answer to Problem 2.7, even though the Intel machine may have a faster clock speed (2.4 GHz vs. 1.2 GHz), that does not necessarily mean the system will perform faster. Different systems are not comparable on clock speed. Other factors such as the system components (memory, buses, architecture) and the instruction sets must also be taken into account. A more accurate measure is to run both systems on a benchmark. Benchmark programs exist for certain tasks, such as running office applications, performing floating-point operations, graphics operations, and so on. The systems can be compared to each other on how long they take to complete these tasks. According to Apple Computer, the G4 is comparable or better than a higher-clock speed Pentium on many benchmarks. 2.9 This representation is wasteful because to represent a single decimal digit from 0 through 9 we need to have ten values. According to Apple Computer, the G4 is comparable or better than a higher-clock speed Pentium on many benchmarks. 2.9 This representation is wasteful because to represent a single decimal digit from 0 through 9 we need to have ten values. Assuming the number of these tubes ON at the same time, then those same tubes could be treated as binary bits. With ten bits, we can represent 2 10 patterns, or 1024 patterns. For integers, these patterns could be used to represent the numbers from 0 through CPI = 1.55; MIPS rate = 25.8; Execution time = 3.87 ns. Source: [HWAN93] -8- 7 2.11 a. ( ) ( ) 10 6 CPI i 1 i CPI A = = I c MIPS A = CPU A = I c CPI A f CPI B = MIPS B = f = 6 CPI A = 90 6 CPI i 1 i I c = CPU B = I c CPI B f = = 0.2 s ( ) f = 6 CPI B = 1024 6 ( ) = = 0.23 s b. Although machine B has a higher MIPS than machine A, it requires a longer CPU time to execute the same set of benchmark programs a. We can express the MIPs rate as: [(MIPS rate)/10 6] = I c /T. So that: I c = T [(MIPS rate)/10 6]. The ratio of the instruction count of the RS/6000 to the VAX is [x 18]/[12x 1] = 1.5. b. For the Vax, CPI = (5 MHz)/(1 MIPS) = 5. For the RS/6000, CPI = 25/18 = From Equation (2.2), MIPS = I c /(T 10 6) = 100/T. The MIPS values are: Computer A Computer B Computer C Program Program Program Program Program Arithmetic mean Rank Harmonic mean Computer A Computer B Computer C Rank -9- 8 2.14 a. Normalized to R: Benchmark Processor R M Z E F H I K Arithmetic mean b. Normalized to M: Benchmark Processor R M Z E F H I K Arithmetic mean c. Recall that the larger the ratio, the higher the speed. Based on (a) R is the slowest machine, by a significant amount. Based on (b), M is the slowest machine, by a modest amount. d. Normalized to R: Benchmark Processor R M Z E F H I K Geometric mean 9 Normalized to M: Benchmark Processor R M Z E F H I K Geometric mean Using the geometric mean, R is the slowest no matter which machine is used for normalization a. Normalized to X: Benchmark Processor X Y Z Arithmetic mean Geometric mean Normalized to Y: Benchmark Processor X Y Z Arithmetic mean Geometric mean Machine Y is twice as fast as machine X for benchmark 1, but half as fast for benchmark 2. Similarly machine Z is half as fast as X for benchmark 1, but twice as fast for benchmark 2. Intuitively, these three machines have equivalent performance. However, if we normalize to X and compute the arithmetic mean of the speed metric, we find that Y and Z are 25% faster than X. Now, if we normalize to Y and compute the arithmetic mean of the speed metric, we find that X is 25% faster than Y and Z is more than twice as fast as Y. Clearly, the arithmetic mean is worthless in this context. b. When the geometric mean is used, the three machines are shown to have equal performance when normalized to X, and also equal performance when normalized to Y. These results are much more in line with our intuition a. Assuming the same instruction mix means that the additional instructions for each task should be allocated proportionally among the instruction types. So we have the following table: Instruction Type CPI Instruction Mix Arithmetic and logic 1 60% Load/store with cache hit 2 18% Branch 4 12% Memory reference with cache miss 12 10% CPI = (2 0.18) + (4 0.12) + (12 0.1) = The CPI has increased due to the increased time for memory access. b. MIPS = 400/2.64 = 152. There is a corresponding drop in the MIPS rate. c. The speedup factor is the ratio of the execution times. Using Equation 2.2, we calculate the execution time as T = I c /(MIPS 10 6) = 11 ms. With 8 processors, each processor executes 1/8 of the 2 million instructions plus the 25,000 overhead instructions. For this case, the execution time for each of the 8 processors is T 8 = 8 = 1.8 ms Therefore we have time to execute program on a single processor Speedup = time to execute program on N parallel processors = = 6.11 d. The answer to this question depends on how we interpret Amdahl's law. There are two inefficiencies in the parallel system. First, there are additional instructions added to coordinate between threads. Second, there is contention for memory access. The way that the problem is stated, none of the code is inherently serial. All of it is parallelizable, but with scheduling overhead. One could argue that the memory access conflict means that to some extent memory reference instructions are not parallelizable. But based on the information given, it is not clear how to quantify this effect in Amdahl's equation. If we assume that the fraction of code that is parallelizable is f = 1, then Amdahl's law reduces to Speedup = N =8 for this case. Thus the actual speedup is only about 75% of the theoretical speedup. -12- 11 2.17 a. Speedup = (time to access in main memory)/(time to access in cache) = T 2 /T 1. b. The average access time can be computed as T = H T 1 + (1 H) T 2 Using Equation (2.8): Speedup = Execution time before enhancement Execution time after enhancement = T 2 T = T 2 = H T 1 + (1 H)T 2 1 (1 H) + H T 1 T 2 C = H T 1 + (1 H) (T 1 + T 2) = This is Equation (4.2) in Chapter 4. Now, Speedup = Execution time before enhancement Execution time after enhancement = T 2 T = T 2 = T 1 + (1 H)T 2 1 (1 H) + H T 1 T 2 In this case, the denominator is larger, so that the speedup is less 12 CHAPTER 3 COMPUTER FUNCTION AND INTERCONNECTION A NSWERS TO Q UESTIONS 3.1 Processor-memory: Data may be transferred from processor to memory or from memory to processor. Processor-I/O: Data may be transferred to or from a peripheral device by transferring between the processor and an I/O module. Data processing: The processor may perform some arithmetic or logic operation on data. Control: An instruction may specify that the sequence of execution be altered. 3.2 Instruction address calculation (iac): Determine the address of the next instruction to be executed. Instruction fetch (if): Read instruction from its memory location into the processor. Instruction operation decoding (iod): Analyze instruction to determine type of operation to be performed and operand(s) to be used. Operand address calculation (oac): If the operation involves reference to an operand in memory or available via I/O, then determine the address of the operand. Operand fetch (of): Fetch the operand from memory or read it in from I/O. Data operation (do): Perform the operation indicated in the instruction. Operand store (os): Write the result into memory or out to I/O. 3.3 (1) Disable all interrupts while an interrupt is being processed. (2) Define priorities for interrupts and to allow an interrupt of higher priority to cause a lower-priority interrupt handler to be interrupted. 3.4 Memory to processor: The processor reads an instruction or a unit of data from memory. Processor to memory: The processor writes a unit of data to memory. I/O to processor: The processor reads data from an I/O device via an I/O module. Processor to I/O: The processor sends data to the I/O device. I/O to or from memory: For these two cases, an I/O module is allowed to exchange data directly with memory, without going through the processor, using direct memory access (DMA). 3.5 With multiple buses, there are fewer devices per bus. This (1) reduces propagation delay, because each bus can be shorter, and (2) reduces bottleneck effects. 3.6 System pins: Include the clock and reset pins. Address and data pins: Include 32 lines that are time multiplexed for addresses and data. Interface control pins: Control the timing of transactions and provide coordination among initiators and targets. Arbitration pins: Unlike the other PCI signal lines, these are not shared lines. Rather, each PCI master has its own pair of arbitration lines that connect it directly to the PCI bus arbiter. Error Reporting pins: Used to report parity and -14- 13 other errors. Interrupt Pins: These are provided for PCI devices that must generate requests for service. Cache support pins: These pins are needed to support a memory on PCI that can be cached in the processor or another device. 64-bit Bus extension pins: Include 32 lines that are time multiplexed for addresses and data and that are combined with the mandatory address/data lines to form a 64-bit address/data bus. JTAG/Boundary Scan Pins: These signal lines support testing procedures defined in IEEE Standard A NSWERS TO P ROBLEMS 3.1 Memory (contents in hex): 300: 3005; 301: 5940; 302: 7006 Step 1: 3005 IR; Step 2: AC Step 3: 5940 IR; Step 4: = 5 AC Step 5: 7006 IR; Step 6: AC Device a. The PC contains 300, the address of the first instruction. This value is loaded in to the MAR. b. The value in location 300 (which is the instruction with the value 1940 in hexadecimal) is loaded into the MBR, and the PC is incremented. These two steps can be done in parallel. c. The value in the MBR is loaded into the IR. 2. a. The address portion of the IR (940) is loaded into the MAR. b. The value in location 940 is loaded into the MBR. c. The value in the MBR is loaded into the AC. 3. a. The value in the PC (301) is loaded in to the MAR. b. The value in location 301 (which is the instruction with the value 5941) is loaded into the MBR, and the PC is incremented. c. The value in the MBR is loaded into the IR. 4. a. The address portion of the IR (941) is loaded into the MAR. b. The value in location 941 is loaded into the MBR. c. The old value of the AC and the value of location MBR are added and the result is stored in the AC. 5. a. The value in the PC (302) is loaded in to the MAR. b. The value in location 302 (which is the instruction with the value 2941) is loaded into the MBR, and the PC is incremented. c. The value in the MBR is loaded into the IR. 4. a. The address portion of the IR (941) is loaded into the MAR. b. The value in location 941 is loaded into the MBR. c. The value in the MBR is stored in location 941 (which is the instruction with the value 2941 placed on the address bus can't access the whole memory. Thus a more complex memory interface control is needed to latch the first part of the address and then the second part (because the microprocessor will end in two steps). For a 32-bit address, one may assume the first half will decode to access a "row" in memory, while the second half is sent later to access -15- 14 a "column" in memory. In addition to the two-step addressing operation, the microprocessor will need 2 cycles to fetch the 32-bit instruction/operand. c. The program counter must be at least 24 bits. Typically, a 32-bit microprocessor will have a 32-bit address bus and a 32-bit program counter, unless double segment registers are used that may work with a smaller program counter. If the instruction register is to contain the whole instruction, it will have to be 32-bits long; if it will contain only the op code (called the op code register) then it will have to be 8 bits long. 3.4 In cases (a) and (b), the microprocessor will be able to access 2 16 = 64K bytes; the only difference is that with an 8-bit memory each access may transfer a byte or a 16-byte word. For case (c), separate input and output instructions are needed, whose execution will generate separate "I/O signals" (different from the "memory signals" generated with the execution of memory-type instructions); at a minimum, one additional output pin will be required to carry this new signal. For case (d), it can support 2 8 = 256 input and 2 8 = 256 output ports and the same number of input and output 16-bit ports; in either case, the distinction between an input and an output port is defined by the different signal that the executed input or output instruction generated Clock cycle = = 125 ns 8 MHz Bus cycle = ns = 500 ns 2 Bytes transferred every 500 ns; thus transfer rate = 4 MBytes/sec Doubling the frequency may mean adopting a new chip manufacturing technology (assuming each instructions will have the same number of clock cycles); doubling the external data bus means wider (maybe newer) on-chip data bus drivers/latches and modifications to the bus control logic. In the first case, the speed of the memory chips will also need to double (roughly) not to slow down the microprocessor. In the second case, the "wordlength" of the memory will have to double to be able to send/receive 32-bit quantities. 3.6 a. Input from the Teletype is stored in INPR. The INPR will only accept data from the Teletype when FGI=0. When data arrives, it is stored in INPR, and FGI is set to 1. The CPU periodically checks FGI. If FGI =1, the CPU transfers the contents of INPR to the AC and sets FGI to 0. When the CPU wants to read more data, it checks FGI. If FGI = 0, the CPU must wait. If FGO = 1, the CPU transfers the contents of the AC to OUTR and sets FGO to 0. The Teletype sets FGI to 1 after the word is printed. b. The process described in (a) is very wasteful. The CPU, which is much faster than the Teletype, must repeatedly check FGI and FGO. If interrupts are used, the Teletype can issue an interrupt to the CPU whenever it is ready to accept or send data. The IEN register can be set by the CPU (under programmer control) 3.7 a. During a single bus cycle, the 8-bit microprocessor transfers one byte while the 16-bit microprocessor transfers two bytes. The 16-bit microprocessor has twice the data transfer rate. b. Suppose we do 100 transfers of operands and instructions. The 16-bit microprocessor requires = 100 bus cycles. The data transfer rates differ by a factor of The whole point of the clock is to define events times on the bus; therefore, we wish for a bus arbitration operation to be made each clock cycle. This requires that the priority signals propagate the length of the daisy chain (Figure 3.26) in one clock period. Thus, the maximum number of masters is determined by dividing the amount of time it takes a bus master to pass through the bus priority by the clock period. 3.9 The lowest-priority device is assigned priority 16. This device would not defer to all the others. However, it may transmit in any slot not reserved by the other SBI devices At the beginning of any slot, if none of the TR lines is asserted, only the priority 16 device may transmit. This gives it the lowest average wait time under most circumstances. Only when there is heavy demand on the bus, which means that most of the time there is at least one pending request, will the priority 16 device not have the lowest average wait time. With a clocking frequency of 10 MHz, the clock period is 10 9 s = 100 ns. The length of the memory read cycle is 300 ns. b. The Read signal begins to fall at 75 ns from the beginning of the third clock cycle (middle of the second half of T 3). Thus, memory must place the data on the bus no later than 55 ns from the beginning of T a. The clock period is 125 ns. Therefore, two clock cycles need to be inserted. 6. From Figure 3.19, the Read signal falls at the beginning of T 2 and kept low for 250 ns a. 5 MHz clock corresponds to a clock period of 200 ns. Therefore, the Write signal has a duration of 150 ns. b. The data remain valid for = 170 ns. c. One wait state a. Without the wait states, the instruction takes 16 bus clock cycles. The instruction requires four memory accesses, resulting in 8 wait states. The instruction, with wait states, takes 24 clock cycles, for an increase of 50%. b. In this case, the instruction takes 26 bus cycles without wait states and 34 bus cycles with wait states, for an increase of 33%. The clock period is 125 ns. One bus read cycle takes 500 ns = 0.5 μs. If the bus cycles repeat one after another, we can achieve a data transfer rate of 2 MB/s. b. The wait state extends the bus read cycle by 125 ns, for a total duration of μs. The corresponding data transfer rate is 1/0.625 = 1.6 MB/s A bus cycle takes 0.25 μs, so a memory cycle takes 1 μs. If both are aligned, the time required is 3 μs. If both are odd-aligned, the time required is 3 μs. If both are odd-aligned, the time required is 3 μs. -16- 3.17 Consider a mix of 100 instructions and operands. On average, they consist of bit items, bit items, and 40 bytes. The number of bus cycles required for the 16-bit microprocessor is (2 20) = 120. For the 32-bit microprocessor, the number required is 100. This amounts to an improvement of 20/120 or about 17% The processor needs another nine clock cycles to complete the instruction. Thus, the Interrupt Acknowledge will start after 900 ns CLK FRAME# AD Address Data-1 Data-2 Data-3 C/BE# Bus Cmd Byte Enable Byte Enable Byte Enable IRDY# TRDY# DEVSEL# Address PhaseAddress PhaseAddress Phase Data Phase Data Phase Data Phase Data Transaction Wait State Wait State Wait State -18- 17 CHAPTER 4 CACHE MEMORY A NSWERS TO Q UESTIONS 4.1 Sequential access: Memory is organized into units of data, called records. Access must be made in a specific linear sequence. Direct access: Individual blocks or records have a unique address based on physical location. Access is accomplished by direct access to reach the general vicinity plus sequential searching, counting, or waiting to reach the final location. Random access: Each addressable location in memory has a unique, physically wired-in addressing mechanism. The time to access a given location is independent of the sequence of prior accesses and is constant. 4.2 Faster access time, greater cost per bit; greater capacity, smaller cost per bit; greater capacity, slower access time. 4.3 It is possible to organize data across a memory hierarchy such that the percentage of accesses to each successively lower level is substantially less than that of the level above. Because memory references tend to cluster, the data in the higher-level memory need not change very often to satisfy memory access requests. 4.4 In a cache system, direct mapping maps each block of main memory into only one possible cache line. Associative mapping permits each main memory block to be loaded into any line of the cache. In set-associative mapping, the cache is divided into a number of sets of cache lines; each main memory block can be mapped into any line in a particular set. 4.5 One field identifies a unique word or byte within a block of main memory. The remaining two fields specify one of the blocks of main memory. These two fields are a line field, which identifies one of the lines of the cache, and a tag field, which identifies one of the blocks that can fit into that line. 4.6 A tag field uniquely identifies a block of main memory. A word field identifies a unique word or byte within a block of main memory. 4.7 One field identifies a unique word or byte within a block of main memory. The remaining two fields specify one of the blocks of main memory. These two fields are a set field, which identifies one of the sets of the cache, and a tag field, which identifies one of the blocks that can fit into that set. 4.8 Spatial locality refers to the tendency of execution to involve a number of memory locations that are clustered. Temporal locality refers to the tendency for a processor to access memory locations that have been used recently. -19- 18.9 Spatial locality is generally exploited by using larger cache blocks and by incorporating prefetching mechanisms (fetching items of anticipated use) into the cache control logic. Temporal locality is exploited by keeping recently used instruction and data values in cache memory and by exploiting a cache hierarchy. A NSWERS TO P ROBLEMS 4.1 The cache is divided into 16 sets of 4 lines each. Therefore, 4 bits are needed to identify the set number. Main memory consists of 4K = 2 12 blocks. Therefore, the set plus tag lengths must be 12 bits and therefore the tag length is 8 bits. Each block contains 128 words. Therefore, 7 bits are needed to specify the word. TAG SET WORD Main memory address = There are a total of 8 kbytes/16 bytes = 512 lines in the cache. The cache consists of 256 sets of 2 lines each. Therefore 8 bits are needed to identify the set number. For the 64-Mbyte main memory, a 26-bit address is needed. Main memory consists of 64-Mbyte/16 bytes = 2 22 blocks. Therefore, the set plus tag lengths must be 22 bits, so the tag length is 14 bits and the word field length is 4 bits. TAG SET WORD Main memory address = There are a total of 8 kbytes/16 bytes = 512 lines in the cache. The cache consists of 256 sets of 2 lines each. Therefore 8 bits are needed to identify the set number. For the 64-Mbyte main memory, a 26-bit address is needed. Main memory consists of 64-Mbyte/16 bytes = 2 22 blocks. Therefore, the set plus tag lengths must be 22 bits, so the tag length is 14 bits and the word field length is 4 bits. TAG SET WORD Main memory address = 24; number of addressable units: 2 24 ; block size: 4; number of blocks in main memory: 2 22 ; number of lines in cache: 4000 hex; size of tag: 22 b. Address length: 24; number of addressable units: 2 24 ; block size: 4; number of blocks in main memory: 2 22 ; number of lines in cache: 2 14 ; size of tag: Block frame size = 16 bytes = 4 doublewords 16 KBytes Number of block frames in cache = 16 Bytes = 1024 Number of sets = Number of block frames Associativity = = 256 sets -20- 19 20 bits 8 4 Tag Set Offset 8 Set 0 20 Tag (20) 4 DWs 4 Decoder Set 1 Comp1 Comp2 Comp3 Set 0 Set 255 Comp4 Set 1 Set 255 Hit Example: doubleword from location ABCDE8F8 is mapped onto: set 143, any line, doubleword 2: 8 F 8 A B C D E (1000) (1111) (1000) Set = 15 for bits 4.7 A 32-bit address consists of a 21-bit tag field, a 7-bit set field, and a 4-bit word field. Each set in the cache includes 3 EU bits and four lines. Each line consists of 4 32-bit words, a valid bit, and a 21-bit tag. 4.8 a. 8 leftmost bits = tag; 3 middle bits = line number; 3 rightmost bits = byte number b. slot 3; slot 6; slot 3; slot 21 c. Bytes with addresses through are stored in the cache d. 256 bytes e. Because two lines with two different memory addresses can be stored in the same place in the cache 4.9 a. The access is according to the following rules with each access to the set: 1. If the access is to L0 or L1, B If the access is to L0, B If the access is to L1, B If the access is to L2 or L3, B If the access is to L2, B0 0. The replacement algorithm works as follows (Figure 4.15): When a line must be replaced, the cache will first determine whether the most recent use was from L0 and L1 or L2 and L3. Then the cache will determine which of the pair of blocks was least recently used and mark it for replacement. When the cache is initialized or flushed all 128 sets of three LRU bits are set to 0. The divides the four lines in a set into two pairs (L0, L1 and L2, L3). Bit B0 is used to select the pair that has been least-recently used. Within each pair, one bit is used to determine which member of the pair was least-recently used. However, the ultimate selection only approximates LRU. Consider the case in which the order of use was: L0, L2, L3, L1. The least-recently used pair is (L2, L3) and the least-recently used member of that pair is L2, which is selected for replacement. However, the least recently used line is L0. Depending on the access history, the algorithm will always pick the least-recently used entry or the second least-recently used entry. -21- 20 The most straightforward way to implement true LRU for a four-line set is to associate a time stamp with each line. When an access occurs, the counter for that block is set to 0; all counters with values lower than the original value for the accessed block are incremented by 1. When a miss occurs and the set is not full, a new block is brought in, its counter is set to 0 and all other counters are incremented by 1. When a miss occurs and the set is full, the block with counter value 3 is replaced; its counter is set to 0 and all other counters are incremented by 1. This approach requires a total of 8 bits. In general, for a set of N blocks, the above approach requires 2N bits. A more efficient scheme can be designed which requires only N(N 1)/2 bits. The scheme operates as follows. Consider a matrix R with N rows and N columns, and take the upper-right triangular portion of the matrix, not counting the diagonal. For N = 4, we have the following layout: R(1,2) R(1,3) R(1,4) R(2,3) R(2,4) R(3,4) When line 1 is referenced, row 1 of R(i,j) is set to 1, and column I of R(j,I) is set to 0. The LRU block is the one for which the row is entirely equal to 0 (for those bits in the row; the row may be empty) and for whose columns are all 1 (for all the bits in the column; the column may be empty). As can be seen for N = 4, a total of 6 bits are required 2 doublewords; associativity K = 2; cache size = 4048 words; C = 1024 block frames; number of sets S = C/K = 512; main memory = 2 s = 2 26 ; number of lines in cache 2 r = 2 6 = 64; size of tag = 20 bits. b. Address format: Tag = 20 bits; Line = 6 bits; Word = 6 bits Number of addressable units = 2 s+w = 2 32 bytes; number of blocks in main memory = 2 s = 2 26 ; number of lines in cache = undetermined; size of tag = 26 bits. c. Address format: Tag = 26 bits; Word = 6 bits; Number of addressable units = 2 s+w = 2 32 bytes; Number of blocks in main memory = 2 s = 2 26 ; Number of lines in set = k = 4; Number of sets in cache = 2 d = 2 17 ; Number of lines in cache = k 2 d = 2 19 ; Size of tag = 9 bits a. Because the block size is 16 bytes and the word size is 1 byte, this means there are 16 words per block set according to the following rules with each access to the set: 1. If the access is to L0 or L1, B If the access is to L0, B If the access is to L1, B If the access is to L2 or L3, B If the access is to L2, B0 0. block 0 through 15 are mapped into sets 0 through 15; blocks 16 through 31 are read into sets 0 through 15; blocks are read into sets 0 through 15; and blocks are read into sets 0 through 15. Because each set has 4 slots, there is no replacement needed through block 63. The last 4 groups of blocks involve a replacement. On each successive passes, 48 blocks are undisturbed, and the remaining 20 must read in. Let T be the access to read 64 words from main memory. If a word is not in the cache, then it can only be ready by first transferring the word from main memory to the cache and then reading the cache. Thus the time to read a block set whose tag matches = 6800T With cache Fetch time = (68) (11T) first pass + (9) (48) (T) + (9) (20) (11T) other passes = 3160T Improvement = 6800T 3160T = 26 4.18 a. Access 63 1 Miss Block 3 Slot 3 Access 64 1 Miss Block 4 Slot 0 Access Hits Access 15 1 Miss Block 0 Slot 0 First Loop Pattern Access 80 1 Miss Block 5 Slot 1 Access Hits Access 15 1 Hit Third Loop Access 16 1 Miss Block 1 Slot 1 Access Hits Access 31 1 Miss Block 1 Slot 1 continues to the Tenth Loop First loop Misses 6 Hits First loop 15-32, Misses 30 Hits Second loop 15-32, Misses 32 Hits Third loop 15-32, Misses 32 Hits Fourth loop 15-32, Misses 32 Hits Ninth loop 15-32, Misses 32 Hits Tenth loop 15-32, Misses 32 Hits Total a. Access 63 1 Miss Block 3 Set 3 Access 64 1 Miss Block 4 Set 0 First Loop Pattern Hits First loop 15-32, Misses 324/348 = b. Access 63 1 Miss Block 3 Slot 1 Access 64 1 Miss Block 4 Set 0 Access Hits Access 15 1 Miss Block 0 Set 0 First Loop Pattern Access 80 1 Miss Block 5 Set 1 Access Hits Access 15 1 Hit Third loop 15-32, Misses 34 Hits Access All bits for the next eight iterations -28- 27 For lines Misses 6 Hits First loop 15-32, Misses 34 Hits Third loop 15-32, Misses 34 Hits Fifth loop 15-32, Misses 34 Hits Seventh loop 15-32, Misses 34 Hits Eighth loop 15-32, Misses 34 Hits Ninth loop 15-32, Misses 34 Hits Tenth loop 15-32, Misses 34 Hits Total = 6 Misses 342 Hits Hit Ratio = 342/348 = a. Cost = C m = = $80 b. Cost = C c = = $800 c. From Equation (4.1) : 1.1 T 1 = T 1 + (H)T 2 (0.1)(100) = (1 H)(1200) H = 1190/a. Under the initial conditions, using Equation (4.1), the average access time is T 1 + (1 - H) T 2 = 1 + (0.05) T 2 Under the changed conditions, the average access time is (0.03) T 2 For improved performance, we must have 1 + (0.05) T 2 > (0.03) T 2 Solving for T 2, the condition is T 2 > 50 b. As the time for access when there is a cache miss become larger, it becomes more important to increase the hit ratio. a. First, 2.5 ns are needed to determine that a cache miss occurs. Then, the requested word is read into cache. Then an additional 2.5 ns are needed to read the requested word. T miss = (15)(5) = 130 ns b. The value T miss from part (a) is equivalent to the quantity (T1 + T2) in Equation (4.1). Under the initial conditions, using Equation (4.1), the average access time is T s = H T 1 + (1 H) (T 1 + T 2) = (0.95)(2.5) + (0.05)(130) = ns Under the revised scheme, we have: -29- 28 and T miss = (31)(5) = 210 ns T s = H T 1 + (1 H) (T 1 + T 2) = (0.97)(2.5) + (0.03)(210) = ns 4.22 There are three cases to consider: Location of referenced word Probability Total time for access in ns In cache Not in cache, but in main memory Not in cache or main memory a. = 12,000,080 So the average access time would be: Avg = (0.9)(20) + (0.06)(80) + (0.04)( ) = ns 4.23 a. Consider the execution of 100 instructions. Under write-through, this creates 200 cache references (168 read references and 32 write references). On average, the read references generate (0.03) 168 = 5.04 read misses. For each read miss, a line of memory must be read in, generating 4 x 4 physical words of traffic. For write misses, a single word is written back, generating 32 words of traffic. For write-through, there are also 32 references for writes. The traffic rate: write-through = byte/instruction Write back = bytes/instruction. For write-back: (0.05) 168 82 + 32 = bytes/instruction For write-back: [(0.07) 168 8] + 32 = bytes/instruction a. 5% miss rate is roughly a crossover point. For a lower miss rate, write-back is superior. For a higher miss rate, write-through is superior a. One clock cycle equals 60 ns, so a cache access takes 120 ns and a main memory access takes 180 ns. The effective length of a memory cycle is ( ) + ( ) = 126 ns. b. The calculation is now ( ) + ( ) = 138 ns. Clearly the performance degrades. However, note that although the memory access time increases by 120 ns, the average access time increases by only 12 ns. 4.25 For a MIPS processor, the average instruction takes 1000 ns to fetch and execute. On average, an instruction uses two operands from memory, each requiring 4 of 600 ns, so the bus utilization for instruction fetch. Bus utilization = the waiting time for other bus requestors, such as DMA devices and other microprocessors. -30- 29 4.26 a. T s = (1 H)T 1 + W(T m T c). T a = T c + (1 H)T m b. Ta = (1 H7) T d + (1 H7 b) = T c + (1 H7) T m + (1 H7 b = 4.277 Ta = (T c (1 H) T m + (1 H 7 4.28 a. miss penalty = x 5 = 0.16 clock cycles. For a line size of one word, average miss penalty = x 5 = 0.16 clock cycles. For a line size of 4 words and the nonburst transfer, average miss penalty = x 20 = 0.22 clock cycles. For a line size of 4 words and the burst transfer, average miss penalty = x 8 = clock cycles. -31- 30 CHAPTER 5 INTERNAL MEMORY A NSWERS TO Q UESTIONS 5.1 They exhibit two stable (or semistable) states, which can be used to represent binary 1 and 0; they are capable of being written into (at least once), to set the state; they are capable of being read to sense the state. 5.2 (1) A memory in which individual words of memory are directly accessed through wired-in addressing logic. (2) Semiconductor main memory in which it is possible both to read data from the memory and to write new data into the memory easily and rapidly. 5.3 SRAM is used for cache memory (both on and off chip), and DRAM is used for main memory. 5.4 SRAMs generally have faster access times than DRAMs. DRAMs are less expensive and smaller than SRAMs. 5.5 A DRAM cell is essentially an analog device using a capacitor; the capacitor can store any charge value within a range; a threshold value determines whether the charge is interpreted as 1 or 0. A SRAM cell is a digital device, in which binary values are stored using traditional flip-flop logic-gate configurations. 5.6 Microprogrammed control unit memory; library subroutines for frequently wanted functions; system programs; function tables. 5.7 EPROM is read and written electrically; before a write operation, all the storage cells must be erased to the same initial state by exposure of the packaged chip to ultraviolet radiation. Erasure is performed by shining an intense ultraviolet light through a window that is designed into the memory chip. EEPROM is a read-mostly memory that can be written into at any time without erasing prior contents; only the byte or bytes addressed are updated. Flash memory is intermediate between EPROM and EEPROM in both cost and functionality. Like EEPROM, flash memory uses an electrical erasing technology. An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM. In addition, it is possible to erase just blocks of memory rather than an entire chip. However, flash memory does not provide byte-level erasure. Like EPROM, flash memory uses only one transistor per bit, and so achieves the high density (compared with EEPROM) of EPROM. 5.8 A0 - A1 = address lines: CAS = column address select; D1 - D4 = data lines. NC = no connect. OE: output enable. RAS = row address select; Vcc = voltage source. Vss = ground. WE: write enable. -32-