Computer architecture and assembly language pdf download

1 Chapter 2 Introduction to Computer Architecture and Language Assembly 2 Computer Architecture Interface between Hardware and the lowest level of software interface between hardware and the lowest level of Software Programmer Language Assembly Vision processor vision of the processor's language 3 basic computer components 3 major components of computer 1. Processor (Central Processing Unit) 1. Processor (Central Processing Unit) 2. Memory 2. Memory 3. Devices in i/O 3. I/O Devices Components are interconnected System Bus. The components are interconnected System Bus.

3 Meeting language Low level, man-readable representation of binary code made by a computer. A low-level, human-read view of a binary code made by a computer. A language that controls primitive operations on binary data. A language that controls primitive operations on binary data. Key operations include moving data, adding, subtracting, comparing, moving, and branching. Key operations include moving data, adding, subtracting, comparing, moving, and branching. 4 Computer Architecture - High-level Language Languages Low Language Language Assembly C, JAVA Word, Excel HHL Complier Assembly Language Machine Language Collector 5 Von Neumann Machine (Stored Program Computer) The general memory system stores both instructions and data. The general memory system stores both instructions and data. Address Data Data Path (Data bus) Address (Address Bus) Central Processor (CPU) Memory Instruction Data 6 Memory Array Data Cells (data/instructions), each cell has a unique addressArray data storage cells (data/instructions), each cell has a unique address address port, port data, control signals. Address port, data port, control signals. Reading cycle: Data at a given memory address is placed in the data bead. Reading cycle: Data at a given memory address is placed in the data bead. Record cycle: Data on the bus data is recorded at the specified location of memory. Record cycle: Data on the bus data is recorded at the specified location of memory. Address Data427 9 6 14 32 Address transmitted from the Reading processor Write 0000 0001 0002 Data transferred between processor and memory 7 CPU (processor) Responds to reading instructions from memory and executing them. Responsible for reading instructions from memory and executing them. Address Path: Used by the processor to provide the memory address of the instruction or memory data. Address Path: Used by the processor to provide the memory address of the instruction or memory data. Data transfer path: Using a processor/memory to transmit data. Data transfer path: Using a processor/memory to transmit data. 8 Base von Neumann The Instruction format consists of the operation code and the operand addressAn instruction consists of an operation operation operation and opera address -May have more than one address, later ... Operation code (op code): determines what the instruction is doing. Operation code (op code): determines what the instruction is doing. Operand address (address): where the operand is located - call the address fieldOperand address (address): where the operand is located - call the address of the field Operation Index Operand address 9 Von Neumann machine in Pseudo code module Von_Neumann I: 0 Repeat Fetch instruction from the memory location I: I No. 1 Run the instruction Forever END Von_Neumann von Neumann machine works in two phases of mode, extraction/ execution cycles. The Von Neumann machine operates in two phases, retrieval/execution cycles. 10 Module Perform instructions To Decode the IF instruction instruction requires data, THEN Fetch data from memory END_IF Perform an operation defined by the IF instruction instruct requires that the data be stored, THEN Save Data in Memory END_IF End Run The Pseudo Code Instruction to perform 11 Information flow between processor and CPU memory C: A'B 5 6 11 Read Write C Instruction Memory 12 Arithmetic and Logic Unit (ALU): calculationArimetric and Logic unit (ALU) ALU: Control Computing Unit (CU): Interprets the Control Unit instruction (CU): interprets the CPU Components Instructions Address Bus Memory Instruction CPU: Data : Bus Control ALU CU Registers 13 Registers: Temporary Storage Registration: Temporary Storage -Program Counter (PC): holds the address of the following instruction to be performed. -Registry of instructions (IR): contains instructions -Data Registers: hold data -Address registers: hold addresses -State Code Registry (CCR): Bits of the flag Updated to reflect the result of the operation Prepared to reflect the result of the operation used to change the flow of the program -MAR, MBR, PSW, etc. (later) Processor components 14 Simple language to describe the operations performed by the processor. A simple language to describe CPU operations. We'll use it to describe the instruction function (4) or M(4) meaning the content of the location of memory 4. Or M(4) means the content of the location of memory 4. M(6) 4 means that the content of the 6 location is 4. M(6) 4 means that the content of the 6 location is 4. M(6) means assigning the number 4 to memory location 6. M(6) means assigning the number 4 to memory location 6. Registration Language Transmission (RTL) 15 Assembly Language Form of the native language of the computer, in which the form of the native language of the computer, in which - the instructions of the machine code are presented to mmmonics - the instructions of the machine code are presented to mmmonics, for example, MOVE, ADD, ADD, SUB - addresses and constants are usually written in symbolic form - addresses and constants, usually written in symbolic form for example, NEXT, BACK_SP, for example, NEXT, BACK_SP 16 MC68000 Assembler The actual symbolic name contains up to 8 letters or number. The actual symbolic name contains up to 8 letters or numbers. The name starts with a letter. The name starts with a letter. TempVa123, TempVa127 recognized TempVa122 by the BuilderTempVa123, TempVa127 recognized TempVa122 as the collector of 17 files created by the assembly of a binary file or object file, recognized by the machine. A binary file or object is recognized by the machine. The listing file contains information about the build of the program. The listing file contains information about the build of the program. If the program is written in multiple files, LINKER needs to tie the object files together before running. If the program is written in multiple files, LINKER needs to tie the object files together before running. Source File Editor Collector Listing File Binary File 18 Language Assembly Program Two types of statementsTwo types of applications 1. Instructions 1. Instructions 2. Collector's Directives 2. Assembling a directive to the Executable InstructionsExecutable Instructions - translated into machine code by a collector - translated into machine code by a collector - tells the machine what to do when executing - tells the machine what to do when performing the 19 Assembly Language Program Assembly DirectiveAssembler directive - tell the builder what to do when assembling the program - tell the builder what to do when assembling the program - not translated into machine code. - not translated into machine code, they are not executed. For example, ELYCRO, D.C., DS, ORG, END, for example, ELYATH, D.C., DS, ORG, END 20 Assembly Language Program, written in 4 columns:Program written in 4 columns: label Instruction (operand) (comment) instruction (operand) - Label: starts in column 1 - Label: starts in column 1 programmer-defined programmer-defined link to the line link to the line $50 is 50 16, %10 000000010, 50 is 50 10.$50 is 50 16% 10 is 00000010, 50 - 50 10. Long word 32-bit, Word 16-bit, Byte 8-bit. Long word 32-bit, Word 16-bit, Byte 8-bit. The line starts with I in my first column is a comment - Ignored Collector Line starts with I in my first column is a comment - the ignored collector 21 Sample programSample program BACK_SP code e'e $08ASCII for backspace DELETE code E'8 $8 $01ASCII to remove the CAR_RET code E'e $0DASCII for the return of transportation ORG $00400Data origin LINE DS. B 64Reserve 64 bytes for the line buffer - Bring the symbol and store it in the LEA LINE Origin BUFFER ORG 2 NEXT BSR GET_DATA CMP. B #BACK_SP,D1 BEQ MOVE_LEFT CMP. B #DELETE,D1 BE' OTMEHA CMP. IN #CAR_RET,D1 BE' EXIT MOVE. B D1,(A2) - BRA NEXT MOVE_LEFT LEA -1(A2),A2 BRA NEXT CANCEL LEA LINE,A2 BRA NEXT GET_DATA MOVE #5,D1 TRAP #15 RTS EXIT STOP No $2700 22 How the picker runs a two-passer-by two pass passer The source of the program is scanned twice before producing the object code LC: Build PC LC Modeling: Build PC Simulation - When the build program is assembled, LC is used to track the place of memory at which the instruction will be in case this instruction is performed. So the machine code can be properly generated from the build code. 23 How does Pass I: Pass I: - Find the source of the character definition program and type them into the Pass II: Pass II character table: - Use a character table built in Pass I and an op-code table to create machine code equivalent to the source of 24 Pass I (Simplified) START (LC) zlt; 0 Fetch Next Instruction END? Pass II Label? Add a label to the W/LC Table symbol as its increments value (LC) respectively Y N Y N 25 Pass II (simplified) START (LC) zlt;- 0 Fetch Next END instruction? STOP Increment (LC) Accordingly Y N Op-Code Lookup Symbol Table Lookup Create Machine Code 26 Sample 1 OPT CRE 2,0000019 A: E'25 3 00001000 ORG $1000 4 000100000004 M: DS. W 2 5 00001004 00001008 N: DC. L OUTPUT 6 00001008 2411 EXIT: MOVE. L (A1),D2 7 0000100A 139A2000 MOVE. B (A2) , (A1,D2) 8 0000100E 06450019 ADDI. W #A.D5 9 0000101012 670000008 BE'S made 10,00001016 90B81004 SUB. L N,D0 11 0000101A 60EC BRA EXIT 12 0000101C 4E722700 DONE: STOP $2700 13,000100 END $1000 Lines: 13, Errors: 0, Warnings: 0. SYMBOL TABLE INFORMATION Symbol-name Type Value Decl Cross Numbers E'E 00000019 2 8. MADE LABEL 0000101C 12 9. EXIT LABELS 00001008 6 5, 11. M LABEL 00001000 4 - NOT used - N LABEL 00001004 5 10. LC Machine Code What we care about in the character table, the build of the code 27 equalizer (E'Uate) Link to the value of the code does not need to change, even if the length and width are changed, the code does not need to change, even if the length and width are changed, It tells the reader how the area is calculated. He tells the reader how the area is calculated. LengthE'30 WidthE'25 AreaE'8Length'Width 28 ORG (ORiGin) Sets Location Counter Value (LC) LC: PCLC Build Modeling: PC 29 DC Assembler Simulation (Determine Constant) Identify and Initiate Variable Skilled. B.S., or. L (byte, word, or longword) Loads constantly in memory in hexadecimalLoads constantly in memory in the hexadecimal 16-bit word should not be stored across a cozy border, for example, at 1001A a 16-bit word should not be stored across a cozy border, for example, in 1001 ORG $00001000 FIRST DC DC. B 10.66 N L $0A1234... 0A0A0A0A0A42 12 34 0A0A0A0A0A0A 001001 001003 001005 001000 001002 001004 Memory Card : : 30 Demonstration - memory is decided by byte, but - data extracted by word, 16 bits. ORG$1000 FIRSTDC. B10,66 SECONDDC. B20 DATADC. L$ABCD STOP-$2700 END$1000 >md 1000 001000 0A 42 14 00 00 00 AB CD 4E 72 27 00 00 00 00. 31 DS (Determine storage) Reserves (distributes) storage space memory is similar to DC, but there are no values stored by DC: adjust values in DS memory locations: reserve memory space for example variables on page 68 32 ORG $001000 TABLE DS. W 256 POINTER_1 DS. L 1 VECTOR_1 DS. L 1 INIT DC. W 0,$FFFF ... ORG $18,000 RECORD LEA ACIAC,A0 MOVE. B #SETUP1,(A0) POINTER_1 01000 011FF 01200 01201 01202 01203 01204 01205 01206 01207 01208 01209 011 20A 0120B 0120B 01200BC 180000 18001 TABLE VECTOR_1 INIT ENTRY 00 FF 41 F9 Example 33 END End of Program 34 Read more about instruction format Is op-code and then address (es) Instruction is op and then address (es) - Address means any address in the System General instructions formats General formats instructions Formats four addresses - Format of four addresses Src1 Src2 Dst NextInstr x y z Src1, Src2: z, Dst: x 35 Read more about the formats of general instructions (Cont'd) General instruction formats (Cont'd) Format of three addresses - format with three addresses (Op-code) Src1 Src2 Dst' Counter program Dst' Use for the following instruction Format with two addresses - Format with two addresses (Op-code) : y, Src2 and Dst: x 68000 uses a format with two addresses 36 Read more about the instructions General Instructions Formats (Cont'd) General instruction formats (Cont'd) - Format with one address Op-code Src1 Battery (AC) is Src2 and DST (AC) - y - format of zero address (Op-code) Using a stack Post-order expression 37 Example and J I and J Next instructions in the NEXT location The following instruction in the next three-address format - Format with three addresses ADD J, K, I; Me and J and K; What's next in the PC Next INSTRUCTION in PC Two-address format - Two address format MOVE J, I; Me AND J ADD K, I; I q K - I How about: ADD K, J MOVE J, I 38 Example I - J and K I - J - One Address FORMAT LOAD J ; AC and J ADD K; AC and J and K I; Format I and zero address, postfix: me and JK - format zero address, mailfix: Me and JK LOAD J; Click J on the LOAD K stack; Click K on the ADD stack Pop and add J and K, result on top of STORE I; Pop stack on top than I 39 More Examples I and J L - M I and J Y K - M I - J and L - L - M I y J (L - M) I y J (L - M) I (L - M) I (J and K) (L) computer architecture and assembly language pdf download