Chapter 4 知识点

CPU:

1. the role of special purpose registers
   1) Program Counter – stores the address of next instruction to be executed
   2) Memory Data Register – stores the data in transit between memory and other registers // holds the instruction before it is passed to the CIR
   3) Current Instruction Register – stores the current instruction being executed
   4) Memory Address Register – stores the address of the memory location which is about to be accessed
2. Describe the role of the registers in the Fetch-Execute (F-E) cycle.
   The **Program Counter (PC)** holds the **address** of the next instruction, and the **contents** are incremented / changed to the next address each cycle
   The **Memory Address Register (MAR)** holds the address to fetch the data (from the PC)
   The **Memory Data Register (MDR)** holds the data at the address in MAR
   The instruction is transferred to **Current Instruction Register (CIR)** for decoding and execution
3. The fetch-execute cycle in register transfer notation.
   MAR <- [PC]
   PC <- [PC] + 1
   MDR <- [[MAR]]
   CIR <- [MAR]
4. System buses
   Data bus
   • Carries data between the processor and memory / carries data that is currently being processed.
   Control bus
   • Transmits signals between the control unit and the other components
   Address bus
   • This uni-directional bus carries signals relating to memory addresses between processor and memory
5. Benefit of increasing the address bus width from 16 bits to 32 bits
   Significant increase in the number of directly addressed memory locations // increases the number of directly addressable memory locations from 2^{16}to2^{32}

6. factors can affect the performance
   1) Number of cores:
      • Each core processes one instruction per clock pulse
      • More/multiple cores mean that sequences of instructions can be split between them
      • … and so more than one instruction is executed per clock pulse // more sequences of instructions can be run at the same time
      • More cores decreases the time taken to complete task
   2) Clock speed:
      • Each instruction is executed on a clock pulse // one F-E cycle is run on each clock pulse
      • ... so the clock speed dictates the number of instructions that can be run per second
      • The faster the clock speed the more instructions can be run per second
   3) Data bus width
      • the width of the data bus determines the number of bits that can be simultaneously transferred
      • increasing the width of the data bus increases the number of bits/amount of data that can be moved at one time (or equivalent)
      •…hence improving processing speed as fewer transfers are needed
      • By example: e.g. double the width of the data bus moves 2x data per clock pulse
   4) address bus width
      increase in the number of directly addressed memory locations
   5) Cache memory
      the higher capacity the more frequently used instructions it can store for fast access
7. Type of interrupts

Tick (✓) **one** box in each row to identify whether each event is an example of a hardware interrupt or a software interrupt.

Event	Hardware interrupt	Software interrupt
Buffer full		✓
Printer is out of paper	✓	
User has pressed a key on the keyboard	✓	
Division by zero		✓
Power failure	✓	
Stack overflow		✓

8. Describe when interrupts are detected in the F-E cycle and how the interrupts are handled.
   1) Detected:
      At the start/end of a FE cycle
   2) Handled:
      i. Priority is checked
      ii. If lower priority than current process continue with F-E cycle
      iii. If higher priority than current process …
      iv. … state of current process is / registers are stored on stack
      v. Location / type of interrupt identified...
      vi. …appropriate ISR is called to handle the interrupt
      vii. When ISR finished, check for further interrupts (of high priority) / return to step 1
      viii. Otherwise load data from stack and continue with process
9. Describe the role of the Arithmetic and Logic Unit (ALU) and Control Unit (CU) in the Von Neumann model.
   ALU:
   • ALU performs arithmetic operations
   • And logical operations / comparisons
   CU:
   • Control Unit sends / receives signals
   • Synchronises operations
   • to control operations // execution of instructions
   • Accept by example e.g. Input output // flow of data
10. Describe the role of the Status Register
   • Status Register is interpreted as independent bits / flags
   • Each flag is set depending on an event
   • An example: addition overflow / result of operation is zero etc.

Assembly Language

1. Mode of addressing
   • Indirect addressing
     The operand is an address, that address holds another address where the data is stored
   • Relative addressing
     the address to be used is an offset number of locations away, relative to the address of the current instruction
   • Indexed addressing

form the address from the given address plus the contents of the index register

• Direct addressing

The operand is the address where the data is stored

• Symbolic addressing

The operand is a word/symbol // A word/symbol represents the memory location/ address

• Immediate addressing

The operand is not an address // the operand is the actual value to be loaded

2. Groups of instructions
   1) Data movement
   2) Input and Output of data
   3) Arithmetic operations
   4) Unconditional and conditional jump instructions
   5) Compare instructions
3. An assembly language program can contain both **macros** and **directives**.

Macro

• A group of instructions given a name // subroutine

• A group of instructions that need to be executed several times within the same program

• The statements are written once and called using the name whenever they need to be executed

• Macro code is inserted into the source file at each place it is called

Directive

• An instruction that directs the assembler to do something

• A directive is not a program instruction

• It is information for the assembler

4. Two-pass Assembler

First pass (steps):

• The assembler scans the assembly language instructions in sequence

• When it meets a symbolic address checks to see if already in symbol table

• If not, it adds it to the symbol table in the symbolic address column

• If it is already in symbol table check if absolute address known

• If the absolute address is known, it is entered in the appropriate cell

• If the absolute address is not known mark / leave as unknown

Second pass

Generation of instruction table

Generation of object code/executable file

Bit Manipulation

1. State the mathematical result of a one-place logical shift to the right on a binary number.

   Division by 2

2. Arithmetic shift:

   Left shift: 符号位不动，右侧补零

   Right shift: 左侧补符号位