Processor Fundamentals

A-Level Computer Science · Vocabulary in context

Word	Example
Von Neumann architecture /vɒn 'nɔɪmən 'ɑːkɪtektʃə/	In the **Von Neumann architecture**, programs and data share one memory.
Immediate Access Store /ɪ'miːdɪət 'ækses stɔː/	The **Immediate Access Store** holds the program that is running.
stored program /stɔːd 'prəʊgræm/	In a **stored program** computer, the program is kept in memory.
processor /'prəʊsesə/	The **processor** runs the program's instructions.
arithmetic and logic unit /ˌærɪθ'metɪk ənd 'lɒdʒɪk 'juːnɪt/	The **arithmetic and logic unit** adds and compares numbers.
ALU /ˌeɪ el 'juː/	The **ALU** does the sums.
operand /'ɒpərænd/	In ADD 5, the number 5 is the **operand**. ADD 5 5
register /'redʒɪstə/	The processor keeps the number it is using in a **register**.
control unit /kən'trəʊl 'juːnɪt/	The **control unit** tells the other parts what to do.
clock speed /klɒk spiːd/	A **clock speed** of 3 GHz means 3 billion cycles each second. 3 GHz 30
fetch-execute cycle /fetʃ 'eksɪkjuːt 'saɪkl/	The processor repeats the **fetch-execute cycle** again and again.
special purpose registers /'speʃl 'pɜːpəs 'redʒɪstəz/	The PC and the MAR are **special purpose registers**. PC MAR

exercises.lol exercises.com.cn

Word	Example
Program Counter /'prəʊgræm 'kaʊntə/	The **Program Counter** holds the address of the next instruction.
Memory Address Register /'meməri ə'dres 'redʒɪstə/	The **Memory Address Register** holds the address to read from.
Memory Data Register /'meməri 'deɪtə 'redʒɪstə/	The **Memory Data Register** holds the data just read.
Current Instruction Register /'kʌrənt ɪn'strʌkʃn 'redʒɪstə/	The **Current Instruction Register** holds the instruction being run.
accumulator /ə'kjuːmjʊleɪtə/	The answer to the sum goes into the **accumulator**.
Status Register /'steɪtəs 'redʒɪstə/	The **Status Register** shows if the last sum overflowed.
flags /flægz/	The processor sets **flags** to show results such as zero or overflow.
bit /bɪt/	A **bit** is a single 0 or 1. 0 1
interrupt /'ɪntərʌpt/	Pressing a key sends an **interrupt** to the processor.
Index Register /'ɪndeks 'redʒɪstə/	The **Index Register** helps the program step through an array.
indexed addressing /'ɪndekst ə'dresɪŋ/	**Indexed addressing** adds the index register to the address.
general-purpose registers /'dʒenərəl 'pɜːpəs 'redʒɪstəz/	A program can use **general-purpose registers** for any data.
register transfer /'redʒɪstə 'trænsfɜː/	A **register transfer** copies data from one register to another.

Word	Example
control bus /kənˈtrəʊl bʌs/	The **control bus** carries signals such as “read” and “write”. “read” “write”
buses /ˈbʌsɪz/	**Buses** are the wires that link the parts of the computer.
address bus /əˈdres bʌs/	The **address bus** carries the memory address.
data bus /ˈdeɪtə bʌs/	The **data bus** carries the data.
word size /wɜːd saɪz/	A 64-bit processor has a **word size** of 64 bits. 64-bit 64 bits
number of cores /ˈnʌmbə ɒv kɔːz/	A larger **number of cores** lets the computer do more at once.
cores /kɔːz/	My laptop’s processor has four **cores**.
amount of RAM /əˈmaʊnt ɒv ræm/	A larger **amount of RAM** lets you open more programs.
RAM /ræm/	Open programs are stored in **RAM**.
page /peɪdʒ/	Memory is split into blocks; each block is a **page**.
cache memory /kæʃ ˈmeməri/	**Cache memory** is faster than RAM. RAM
cache /kæʃ/	The **cache** keeps data the processor needs often.
secondary storage /ˈsekəndəri ˈstɔːrɪdʒ/	A hard disk is **secondary storage**.
port /pɔːt/	Plug the mouse into a USB **port**. USB
peripheral /pəˈrɪfərəl/	A printer is a **peripheral**.

exercises.lol exercises.com.cn

Word	Example
register transfer notation /ˈredʒɪstə ˈtrænsfɜː nəʊˈteɪʃn/	In **register transfer notation**, MAR ← PC copies the PC into the MAR. MAR ← PC PC MAR
interrupt service routine /ˈɪntərʌpt ˈsɜːvɪs ruːˈtiːn/	The **interrupt service routine** deals with the key press.
interrupt register /ˈɪntərʌpt ˈredʒɪstə/	The **interrupt register** shows which interrupt is waiting.
stack /stæk/	The last item put on the **stack** is the first one taken off.
assembly language /əˈsemblɪ ˈlæŋgwɪdʒ/	**Assembly language** uses short words such as ADD and LDM. ADD LDM
machine code /məˈʃiːn kəʊd/	The processor runs **machine code**: only 0s and 1s. 0 1
mnemonics /nɪˈmɒnɪks/	ADD and SUB are **mnemonics**. ADD SUB
assembler /əˈsemblə/	The **assembler** turns assembly language into machine code.
symbol table /ˈsɪmbl ˈteɪbl/	The **symbol table** stores each label and its address.
label /ˈleɪbl/	The **label** LOOP marks the start of the loop. LOOP
forward references /ˈfɔːwəd ˈrefrənsɪz/	**Forward references** use a label before it is defined.
opcode /ˈɒpkəʊd/	In ADD 5, ADD is the **opcode**. ADD 5 ADD
addressing mode /əˈdresɪŋ məʊd/	The **addressing mode** tells the processor how to find the operand.

Word	Example
immediate addressing /ɪˈmiːdɪət əˈdresɪŋ/	In **immediate addressing**, the operand is the value itself.
direct addressing /daɪˈrekt əˈdresɪŋ/	In **direct addressing**, the operand is the address of the value.
indirect addressing /ɪndaɪˈrekt əˈdresɪŋ/	In **indirect addressing**, the address holds another address.
relative addressing /ˈrelətɪv əˈdresɪŋ/	**Relative addressing** counts from the current instruction.
logical shift /ˈlɒdʒɪkl ʃɪft/	A **logical shift** moves every bit and fills the gap with 0. 0
cyclic shift /ˈsaɪklɪk ʃɪft/	In a **cyclic shift**, the bit that falls off comes back at the other end.
mask /mæsk/	Use a **mask** to keep only the last four bits.

exercises.lol exercises.com.cn