

UNIVERZITA KONŠTANTÍNA FILOZOFA V
NITRE
FAKULTA PRÍRODNÝCH VIED

Informačné technológie

Vypracované otázky na štátnu skúšku

Obsah

1	Architektúra RISC a CISC	2
1.1	CPU	2
1.2	RISC a CISC	5
2	Multiprocesorový počítač a multiprocesorový systém	7
3	Bezpečnosť technických prostriedkov, NAS servery	8
4	Riadiace počítače a ich využitie v priemyselnej technike	9
5	Cloud computing	10
6	Distribované systémy a distribuované výpočty	11
7	Migrácia procesov	12
8	Medziprocesová komunikácia	13
9	Synchronizácia hodín	14
10	Bezdrôtové senzorové siete	15
11	Charakteristika expertných systémov	16
12	Reprezentácia znalostí	17
13	Odovodzovací mechanizmus	18
14	Zložky znalostného systému a tvorba znalostných systémov	19
15	Rozhodovacie stromy	20
16	Základné pojmy v kybernetike	21
17	Kybernetický systém, riadenie dynamických systémov	22
18	Typy kybernetických systémov a prvky ich riadenia	23
19	Prostriedky automatického riadenia v kybernetike	24
20	Automatizované systémy	25

1. Architektúra RISC a CISC

Korpus: *Definícia pojmu procesor, rozdiel medzi architektúrami RISC a CISC, príklady spracovávanía inštrukcií, využitie architektúry CISC v súčasnosti, výhody a nevýhody malého počtu inštrukcií. Pri uvádzaní rozdielov medzi jednotlivými architektúrami je potrebné, aby študent vedel rozlíšiť od seba nielen architektúru RISC a CISC, ale aby vedel popísať hlavne prečo vznikla nová architektúra RISC a čo ňou je možné dosiahnuť z hľadiska používania tzv. inštrukčných mixov.*

1.1. CPU

Central Processing Unit, alebo procesor, interpretuje, vykonáva inštrukcie a spracúva dáta programu vo forme strojového kódu. V dnešnej dobe je zvyčajne realizovaný vo forme mikroprocesora. Historicky boli CPU implementované ako diskkrétne komponenty a viacero malých integrovaných obvodov na jednej či viacerých doskách. Mikroprocesory sú CPU vyrábané zvyčajne na jednom integrovanom obvode. Zmenšenie veľkosti CPU prispieva k rýchlejšiemu prepínaniu, vďaka fyzickým faktorom, ako napríklad zníženej parazitickéj kapacitancii. Hoci komplexnosť, veľkosť a výkon CPU sa od roku 1950 výrazne zmenili, základný dizajn a funkčnosť sa zachovali. Skoro všetky bežné CPU sa dajú pomerne presne popísať Von Neumannovou schémou.

Práca CPU Základnou funkciou CPU je vykonávať sekvenciu uložených inštrukcií¹. Vykonávané inštrukcie bývajú uložené v nejakej forme **počítačovej pamäti**. Skoro všetky CPU nasledujú fetch, decode a execute kroky v ich operácii, ktoré sa spolu nazývajú **inštrukčným cyklom**.. Po vykonaní inštrukcie sa celý proces zopakuje s nasledujúcou inštrukciou označenou inkrementovanou hodnotou v registri **program counter**. Pokiaľ sa vykonala inštrukcia **skoku**², program counter sa nastaví na cieľovú inštrukciu a vykonávanie programu prebieha ďalej normálne. Vo viac zložitých CPU sa môže vykonávať inštrukčný cyklus na viacerých inštrukciách súčasne. Niektoré inštrukcie (napríklad *jump*) nemanipulujú dátami, ale zato upravujú program counter. Tým umožňujú praktiky, napr. cykly³, podmienené vykonávanie inštrukcií⁴ a existenciu funkcií.

Fetch Prvý krok zaisťuje získanie inštrukcie z pamäte programu. Adresa inštrukcie je určená hodnotou v **PC** (program counter). Keď je inštrukcia získaná, PC je inkrementovaný dĺžkou inštrukcie tak, aby obsahoval adresu nasledujúcej inštrukcie. Keďže inštrukcie bývajú uložené v relatívne pomalých pamätiach (RAM, swapfile na disku), existuje riziko, že procesor bude na inštrukciu musieť čakať. Tento problém sa v moderných architektúrach rieši použitím cache a pipeline architektúr.

Decode V tomto kroku sa inštrukcia konvertuje na signály, ktoré ovládajú ostatné časti CPU, za použitia **dekódera inštrukcií**⁵. Spôsob akým sa toto deje je definovaný architektúrou inštrukčnej sady⁶. Zvyčajne jedna skupina bitov, zvaná opcode, označuje, ktorá operácia sa má vykonať, kým ostatné polia poskytujú dodatočné informácie dôležité pre operáciu, napr. operandy. Tieto operandy môžu byť konštatné hodnoty, alebo lokácie hodnoty, ktorá môže byť buď register procesoru, alebo adresa pamäte. V niektorých dizajnoch CPU býva dekódér inštrukcií implementovaný ako nemeniteľný obvod. V iných sa na dekódovanie in-

¹program

²jump

³for, while loop

⁴if, switch-case

⁵Instruction decoder

⁶ISA, Instruction Set Architecture

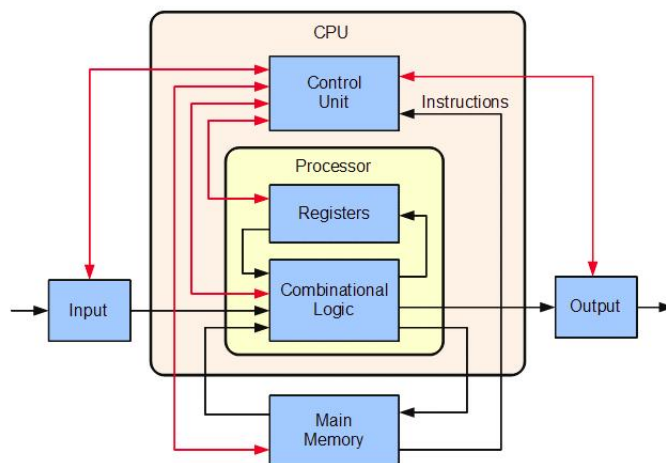
štrukcií používa mikrogram, ktorý môže byť prepísaný, čím sa dá upraviť spôsob, ako CPU dekoduje inštrukcie. Takto sa napríklad riešia zraniteľnosti, napr. Spectre a Meltdown.

Execute Posledným krokom inštrukčného cyklu sa vykonávajú inštrukcie. V závislosti od architektúry CPU, toto môže byť spravené jednou akciou, alebo ich sekvenciou. Počas každej akcie sa rôzne časti CPU elektricky prepoja, aby mohli vykonať celú, či časť operácie v závislosti od tikotu hodín. Častokrát sa výsledny zapisujú do vnútorného registra CPU pre rýchly prístup z nasledujúcich inštrukcií. Pokiaľ sa k nim v skorej dobe nebude pristupovať, zapisujú sa do pomalšej, ale lacnejšej hlavnej pamäti. Napríklad ak sa má vykonať inštrukcia sčítania, vstupy do aritmeticko-logickej jednotky⁷ sú prepojené s párom operandov, výsledok sa objaví na výstupe, ktorý je pripojený na úložisko (register, alebo pamäť). Keď nastane tik hodín, výsledok sa presunie do úložiska a pokiaľ je výsledok väčší ako veľkosť výstupného slova ALU, nastaví sa arithmetic overflow flag.

Štruktúra CPU V CPU sú na pevno definované základné operácie, ktoré dokáže vykonávať, tzv. **inštrukčná sada**⁸. Medzi takéto operácie patrí napríklad sčítanie, odčítanie, porovnanie dvoch čísel či skok do inej časti programu. Každá základná operácia je reprezentovaná kombináciou bitov, ktorá sa nazýva **opcode** strojového jazyka. Program v strojovom jazyku je zoskupenie inštrukcií v strojovom jazyku, ktoré CPU vykonáva. Matematické operácie pre každú inštrukciu sa vykonávajú kombinatorickým logickým obvodom, ktorý sa označuje ako Aritmeticko logická jednotka, alebo ALU. Základnými časťami procesora teda sú:

- Aritmeticko-logická jednotka
- Riadiaca jednotka⁹

Viac komplexné mikroprocesory súčasnosti obsahujú aj tzv. **jednotku pridelovania pamäte**, alebo **MMU**¹⁰. Jednoduché procesory, napr. mikrokontroléry zvyčajne neobsahujú MMU.



Obr. 1.1. Základná schéma CPU. Čierne čiary označujú tok dát, červené tok príkazov, šípky smer.

⁷ALU, Arithmetic Logic Unit

⁸instruction set

⁹Control Unit

¹⁰Memory Management Unit

ALU Aritmeticko logická jednotka je kombinačný digitálny elektrický obvod, ktorý vykonáva aritmetické a bitové operácie na celočíselných binárnych číslach. Tento koncept navrhol John von Neumann v 1945 v správe o základoch nového počítača EDVAC. Na vstupe do ALU sú dáta, zvané operandy a kód operácie, ktorá má byť vykonaná. Na výstupe je výsledok vykonanej operácie. Vo viacerých dizajnoch má ALU aj stavové vstupy a výstupy, ktoré nesú informáciu o predchádzajúcej, alebo súčasnej operácii medzi ALU a externými stavovými registrami.

CU Ďalšou časťou von Neumannovej architektúry je riadiaca jednotka ¹¹. Je to komponent CPU, ktorý riadi činnosť procesora. Hovorí pamäti, ALU, a vstupno-výstupným zariadeniam ako majú reagovať na inštrukcie programu. Riadi operácie ostatných jednotiek poskytovaním časových a riadiacich signálov. CU riadi tok dát medzi CPU a ostatnými zariadeniami. Jednotlivé inštrukcie prechádzajú cez CU, ktorá ich dekoduje na niekoľko sekvenčných krokov, ktoré riadia a koordinujú vnútorné mechanizmy CPU pre správnu manipuláciu dát.

Registre Registre sú rýchlo dostupné oblasti prístupné procesoru. Zvyčajne obsahujú malé, no veľmi rýchle úložisko, hoci môžu mať aj iné, špecifické hardvérové funkcie. Pamäť v registri býva typicky adresovateľná mechanizmami oddelenými od hlavnej pamäte, no v istých prípadoch môžu mať pridelenú adresu pamäte. Skoro vo všetkých architektúrach sa dáta načítavajú z väčších pamätí do registrov, odkiaľ sa používajú v aritmetických operáciách. Výsledky sa bežne ukládajú do hlavnej pamäti. Poznáme rôzne kategórie registrov, líšiace sa účelom, použitím a množstvom bitov, ktoré môžu uchovávať.

- Používateľsky-prístupné registre

- Dátové registre
- Adresové registre
- GPR ¹² - môžu uchovávať dáta i adresy.
- Stavové registre
- Floating-point registre
- Registre konštánt - napr. 0, 1, π
- Vektorové registre - uchovávať dáta pre SIMD¹³
- SPR¹⁴ - uchovávať stav programu. Patrí sem program counter, status register.
- MTRR¹⁵ - sada registrov, ktoré poskytujú systému riadenie toho, ako sú prístupy CPU k pamäti cachované.

- Vnútorné registre

- Register inštrukcií - drží informáciu o momentálne vykonávanej inštrukcii.
- Registre oddelené od CPU súvisiace so získavaním informácie z RAM
 - * Memory buffer register (MBR)
 - * Memory data register (MDR)
 - * Memory address register (MAR)

- Architektúrne registre

¹¹Control Unit

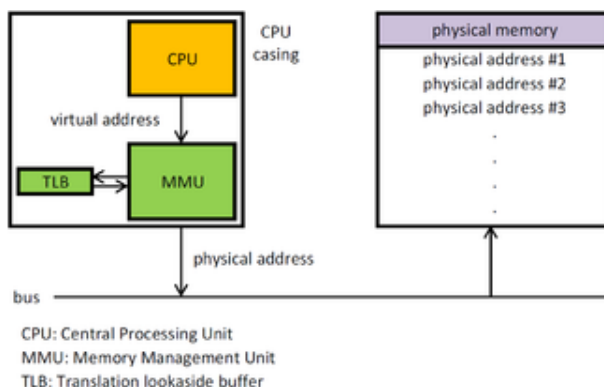
¹²General-purpose registers

¹³Single Instruction, Multiple Data

¹⁴Special-purpose registers

¹⁵Memory Type Range Registers

Jednotka pridelovania pamäte Memory Management Unit je komponent, skrz ktorý prechádzajú všetky pamäťové referencie a ktorý predovšetkým vykonáva preklad adres virtuálnej pamäte na fyzické adresy. Spravidla býva implementovaná ako súčasť CPU, avšak môže sa vyskytovať aj ako samostatný integrovaný obvod. MMU sa stará o manažment virtuálnej pamäte, tzn. riadi ochranu pamäte, riadenie cache, a v jednoduchších architektúrach aj tzv. "bank switching"¹⁶ Moderné MMU typicky delia adresný priestor na stránky, ktoré majú veľkosť v násobkoch 2, zvyčajne pár kilobytov, hoci môžu byť aj väčšie.



Obr. 1.2. Schéma práce MMU.

1.2. Harvardská a Princetonská architektúra

1.3. RISC a CISC

ISA Architektúra inštrukčnej sady je abstraktný model počítača. Realizácia ISA sa nazýva jej implementáciou. ISA definuje všetko čo musí programátor strojového jazyka vedieť na programovanie počítača. To, čo jednotlivé ISA definujú sa líši prípad od prípadu. Vo všeobecnosti definuje podporované dátové typy, stavy a ich sémantiky, inštrukčnú sadu a I/O model. ISA sa líši od mikroarchitektúry, čo je sada dizajnových techník použitých v konkrétnom procesore na implementáciu inštrukčnej sady. Procesory rôznych mikroarchitektúr môžu mať spoločnú inštrukčnú sadu. Napríklad procesory Intel Pentium a AMD Athlon implementujú takmer totožné verzie x86 inštrukčnej sady, hoci majú radikálne odlišný vnútorný dizajn. Dve hlavné kategórie ISA sú CISC¹⁷ a RISC¹⁸. Hoci sa jedná o protichodné koncepty, rozdiely medzi nimi sú však občas ťažko rozpoznateľné, keďže napríklad niektorí autori opisujú mikrokontroléry PIC ako RISC, iní zas ako CISC. Taktiež procesory x86, hoci sú CISC, mnoho z nich vnútorne používa mikrokód na konverziu komplexných inštrukcií na sekvenciu jednoduchších, ktoré potom vykoná. Hovorí sa preto, že hoci x86 **inštrukčná sada** je CISC, moderná x86 **architektúra** je RISC.

CISC	RISC
Dôraz na hardvér	Dôraz na softvér
Aj viac-taktové komplexné inštrukcie	Len jedno-taktové redukované inštrukcie
"LOAD" a "STORE" vstavané v inštrukciách	"LOAD" a "STORE" ako nezávislé inštrukcie
Menšia veľkosť kódu, viac cyklov za sekundu	Menej cyklov za sekundu, väčšia veľkosť kódu
Tranzistory ukládajú komplexné inštrukcie	Viac tranzistorov v pamäťových registroch

¹⁶Technika ktorou sa v starších osem bitových počítačoch rozširovala pamäť nad rámec normálne adresovateľného priestoru. Bloky pamäte, ku ktorej bolo potrebné pristupovať len raz, napr. pri boote systému sa mohli vymeniť s prázdnu časťou operačnej pamäte.

¹⁷Complex Instruction Set Computer

¹⁸Reduced Instruction Set Computer

CISC Starší z typov. Jednotlivé inštrukcie sú schopné vykonávať viacero nízko-úrovňových operácií¹⁹, alebo viackrokových operácií, či adresových módov počas jednej inštrukcie. Keďže sa tento názov začal používať retroaktívne po nástupe RISC procesorov, slúži ako kolektívny názov pre všetko, čo nie je RISC, či už sa jedná o komplexné sálové počítače, po najjednoduchšie mikrokontroléry, kde ukladanie a načítanie z pamäti nie sú oddelené od aritmetických inštrukcií. Z tohto dôvodu môže byť paradoxne moderný RISC procesor výrazne komplexnejší ako napríklad moderný CISC mikrokontroler, obzvlášť z hľadiska komplexnosti elektrických obvodov, ale aj počtom inštrukcií či ich zložitosti. Jediná typicky odlišujúca charakteristika je, že väčšina RISC dizajnov používa jednotnú inštrukčnú dĺžku pre skoro všetky svoje inštrukcie a striktne oddeluje inštrukcie load/store od ostatných.

Výhody CISC Predtým, než sa uplatnila dizajnová filozofia RISC, mnoho počítačových architektov sa snažilo zmenšiť takzvanú "sémantickú priepasť", tzn. navrhovať inštrukčné sady priamo podporujúce programovacie konštrukty vyššej úrovne, napr. volania procedúr, cykly a komplexné adresné režimy, umožňujúce dátovú štruktúru a prístupy k poliam skombinovať do jednotlivých inštrukcií. Inštrukcie sú typicky vysoko kódované, aby sa navýšila hustota kódu. Takáto kompaktnosť vedie k menším programom a menšiemu počtu prístupov do hlavnej (pomalej) pamäti, čo v dobe vzniku²⁰ viedlo k veľkému šetreniu na cene pamäte a úložiska, a aj rýchlosti výpočtov. Taktiež to viedlo k lepšej produktivite programátorov aj v assembly, keďže vysoko úrovňové jazyky ako Fortran a Algol neboli vždy dostupné, alebo vhodné.

Pridávanie inštrukcií V 70. rokoch sa predpokladalo, že pridanie ďalších inštrukcií by mohlo navýšiť výpočtový výkon. Kompilátory boli aktualizované, aby využili tieto inštrukcie a takýto prístup sa používa v procesoroch dodnes, keďže prístupy do hlavnej operačnej pamäte sú stále niekoľko rádovo pomalšie ako činnosť jadra CPU.

Nevýhody dizajnu Kým úmyslom navyšovania počtu inštrukcií bolo zvýšiť výkon a umožniť vysoko úrovňovým jazykovým konštruktom vyjadrenie menším počtom inštrukcií, nie vždy sa to takto darilo. Architekti občas predizajnovali inštrukcie strojového jazyka, zahrňujúc funkcie, ktoré nemohli byť efektívne implementované na hardvéri.

RISC Táto architektúra umožňuje počítačom potrebovať menej cyklov na inštrukciu, ako porovnateľný CISC. Koncept RISC sa datuje do 80. rokov minulého storočia. O jeho popularizáciu sa zaslúžili najmä dva projekty zo Standfordskej univerzity a University of California, Berkeley. Standfordská MIPS architektúra sa neskôr úspešne skomercializovala, kým architektúra vyvinutá v Berkeley, RISC, dala názov celému konceptu a ujala sa ako SPARC. Rôzne varianty RISC dizajnu zahrňujú ARC, Alpha, Am29000, ARM, Atmel AVR, Blackfin, i860, i960, M88000, MIPS, PA-RISC, Power ISA (including PowerPC), RISC-V, SuperH, and SPARC.

Hoci sa podobné koncepty a implementácie objavovali už skôr, napr. CDC 6600 navrhnuté Seymourom Crayom v 1964, Michael J. Flynn považuje za prvý RISC systém IBM 801. Do vývoja RISC sa dala aj DARPA skz svoj VLSI²¹ program. Berkeley RISC bol navrhnutý tak, aby získal náskok vo výkone použitím pipeliningu a agresívnym používaním techniky register windowing. V tradičnom CPU je len malý počet registrov a program k nim môže kedykoľvek pristupovať. V CPU s oknami registrov je obrovský počet registrov, ale programy môžu využívať naraz len malý počet z nich. Program, ktorý sa limituje na osem registrov na procedúru môže vykonávať veľmi rýchle volania. Volanie jednoducho "posunie" okno nadol o osem, k sade registrov, ktoré procedúra vyoužíva a pri návrate posunie

¹⁹Napr. načítanie z pamäte, aritmetické operácie, či uloženie do pamäti.

²⁰60. roky 20. storočia

²¹Very-Large-Scale Integration

okno späť. RISC-I procesor, vyvinutý v 1982 pozostával z 44 420 tranzistorov (oproti cca 100 000 v CISC procesoroch tej doby), mal len 32 inštrukcií, no vo výkone prekonal všetky ostatné jednočipové dizajny. Neskôr ho predčili procesorom RISC-II s 40 760 tranzistormi a 39 inštrukciami, ktorý bol 3x rýchlejší ako RISC-I.

V 2010 sa objavila nová, open source ISA RISC-V, vyvinutá v Berkeley pre výskumné účely a ako slobodná alternatíva voči proprietárnym ISA. V súčasnosti sa objavujú prvé implementácie vo forme SiFive procesorov.

ZISC

2. Multiprocesorový počítač a multiprocesorový systém

Korpus: Definícia, rozdelenie podľa kategórií, stavba vnútornej pamäte. Využitie paralelných počítačov z hľadiska Flynnovej klasifikácie v súčasnosti, paralelizmus v procesoroch (porovnanie jedno a viacjadrovej architektúry, viacprocesorovej architektúry), Amdahlov zákon. Je potrebné, aby študent vedel preukázať opodstatnenosť využitia paralelizmu v súčasných typoch procesorov a počítačových systémov napr. formou Grid technológie, alebo iného spôsobu spracovania veľkého počtu požiadaviek v relatívne krátkom čase.

3. Bezpečnosť technických prostriedkov, NAS servery

Korpus: Bezpečnosť technických prostriedkov a ukladania dát na Slovensku, problematika DAS (Direct Attached Storage), NAS (Network Attached Storage) a SAN (Storage Area Network) a Fault Tolerant systémy. Študent musí vedieť definovať výhody nasadenia SAN, z čoho pozostáva dnešné typické SAN riešenie.

4. Riadiace počítače a ich využitie v priemyselnej technike

Korpus: Definícia a architektúra riadiaceho počítača, základné požiadavky kladené na riadiace počítače z hľadiska ich implementácie do výrobného procesu. Je potrebné aby študent vedel definovať riadiaci počítač z pohľadu jeho vnútornej architektúry a zároveň vedel uviesť najbežnejšie príklady riadiacich počítačov používaných v súčasnosti v praxi (minimálne 3) a porovnal ich z pohľadu počtu a možnosti využitia analógových/digitálnych vstupov a výstupov, riadiacej jednotky, pamäte a pod.

5. Cloud computing

Korpus: Definícia pojmu cloud computing, hlavné charakteristiky a modely služieb Cloud computingu (Software as a Service (SaaS), Platforma as a Service (PaaS), Infrastructure as a Service (IaaS)), praktický príklad použitia cloudu, výhody, nevýhody a riziká využitia cloudu.

6. Distribuované systémy a distribuované výpočty

Korpus: Definícia distribuovaných systémov, úrovne transparentnosti v distribuovaných systémoch, architektúry a modely distribuovaných systémov.

7. Migrácia procesov

Korpus: Stavové modely procesov, definícia procesu a vlákna, mapovanie vlákien, migračné stratégie.

8. Medziprocesová komunikácia

Korpus: Základné princípy medziprocesovej komunikácie, synchronná a asynchronná komunikácia, vzdialené volanie procedúr, sockety, MPI komunikácia.

9. Synchronizácia hodín

Korpus: Dôvod synchronizácie hodín, mechanizmy šírenia časového signálu, logické hodiny (Lamportov algoritmus), synchronizácia fyzických hodín (Cristianov algoritmus), protokol NTP, Berkeley algoritmus.

10. Bezdrôtové senzorové siete

Korpus: Architektúry bezdrôtových senzorových sietí, lokácia v bezdrôtových senzorových sieťach, praktické využitie senzorových sietí.

11. Charakteristika expertných systémov

Korpus: Znalostný systém, silné a slabé metódy riešenia problémov, atribúty znalostí (poznatkov), charakteristické vlastnosti znalostných systémov, výhody oddelenej reprezentácie poznatkov, ohraničenia - problémy znalostných systémov, typy problémov pre znalostné systémy, základné architektúry znalostných systémov, minimálna kostra znalostného agenta.

12. Reprezentácia znalostí

Korpus: Reprezentácia poznatkov, požiadavky na jazyk pre reprezentáciu poznatkov, logika (výroková, predikátová), pravidlá, vizualizácia pravidiel pomocou AND/OR grafu, sémantické siete, rámce, konceptuálne mriežky.

13. Odvodzovací mechanismus

Korpus: Odvodzovací mechanismus, vnútorný a vonkajší pohľad na odvodzovací mechanismus, usudzovanie s pravidlami, stratégie riešenia konfliktov, dopredné a spätné zreťazenie, optimalizácia tvorby aplikovateľných pravidiel, usudzovanie so sémantickými sieťami, usudzovanie s rámcami.

14. Zložky znalostného systému a tvorba znalostných systémov

Korpus: Vstupno-výstupný modul, vysvetľovací modul, životný cyklus ZS, etapy získavania znalostí (identifikácia a analýza problému, konceptualizácia, formalizácia, implementácia, validácia), účastníci v procese tvorby ES, techniky získavania znalostí, spoľahlivosť a neúplnosť znalostí, interview, voľné asociácie, monitorovanie, komentovanie postupu, dialógy, príznakové mriežky a postupy pri jej vytváraní.

15. Rozhodovacie stromy

Korpus: Extrakcia znalostí z údajov, rozhodovacie stromy, rozhodovací strom ako klasifikátor, algoritmus tvorby rozhodovacieho stromu, nájdenie vhodnej vlastnosti pre delenie, tréovanie rozhodovacieho stromu, prepis rozhodovacieho stromu na pravidlá

16. Základné pojmy v kybernetike

Korpus: Definícia pojmu kybernetika, historický prehľad, základné pojmy, štruktúra kybernetiky, teoretická, aplikovaná a technická kybernetika.

17. Kybernetický systém, riadenie dynamických systémov

Korpus: Definícia systému riadenia, riadený a riadiaci systém, regulačný obvod, regulovaná sústava, regulátor.

18. Typy kybernetických systémov a prvky ich riadenia

Korpus: Otvorený - uzavretý, lineárny - nelineárny, spojitý – nespojitý, jednoduchý – rozvetvený, jednoparametrový –viacparametrový, regulácia, cieľ riadenia, informáciu o stave okolia a riadeného systému, pôsobenie na objekt (vlastné riadenie), algoritmus riadenia (pravidlo udávajúce, aký spôsobom je možné dosiahnuť cieľ na základe informácie o stave okolia a riadeného systému).

19. Prostriedky automatického riadenia v kybernetike

Korpus: Získavanie, prenos, úprava a spracovanie informácií, akčné a riadiace systémy

20. Automatizované systémy

Korpus: Riadenie výrobných strojov, robotov a výrobných systémov. Automatizácia nevýrobnej sféry - inteligentné domy, zabezpečovanie, informačné systémy, trendy automatizácie.