Co je správně?
 :r1 Jeden bit má osm bajtů.
 :r2 Jeden bajt má osm bitů.
 :r3 Jeden bajt je složen ze dvou nebo čtyř slov.
:r2 ok
--
Nejmenší adresovatelná jednotka paměti je
 :r1 kapacita místa v paměti, které má vlastní adresu.
 :r2 nejmenší hodnota adresy v paměti.
 :r3 nejmenší číslo, které lze do paměti uložit.
:r1 ok
--
Nejmenší adresovatelná jednotka paměti typicky je
 :r1 1 bit
 :r2 8 bitů
 :r3 16 bitů
:r2 ok
--
1 KB je
 :r1 1000 B
 :r2 1048 b
 :r3 1024 B
:r3 ok
--
2¹⁰ bajtů je
 :r1 1 KB
 :r2 128 KB
 :r3 512 KB
 :r4 1 MB
:r1 ok
--
2¹⁶ bajtů je
 :r1 24 KB
 :r2 32 KB
 :r3 64 KB
 :r4 128 KB
:r3 ok
--
2²⁰ bajtů je
 :r1 256 KB
 :r2 512 KB
 :r3 1 MB
 :r4 2 MB
 :r5 4 MB
:r3 ok
--
2³² bajtů je
 :r1 2 MB
 :r2 4 MB
 :r3 1 GB
 :r4 2 GB
 :r5 4 GB
:r5 ok
--
Adresový registr obsahuje 4 bity. Kolik je schopen namapovat (zaadresovat)
adres?
 :r1 4
 :r2 8
 :r3 10
 :r4 16
 :r5 20

:r4 ok
--
Paměť o maximální kapacitě 1 M adresovatelných míst musí mít adresovací sběrnici širokou právě
 :r1 32 bitů
 :r2 21 bitů
 :r3 20 bitů
 :r4 30 bitů
:r3 ok
--
Paměť o maximální kapacitě 1 G adresovatelných míst musí mít adresovací sběrnici širokou právě
 :r1 32 bitů
 :r2 21 bitů
 :r3 20 bitů
 :r4 30 bitů
:r4 ok
--
Jaká je správná posloupnost seřazená podle velikosti uchovávané informace od nejmenší po největší?
 :r1 bit, slovo, bajt
 :r2 bit, bajt, slovo
 :r3 bajt, slovo, bit
 :r4 bajt, bit, slovo
 :r5 slovo, bajt, bit
:r2 ok
--
Paměť RAM
 :r1 se řadí mezi paměti se sekvenčním přístupem
 :r2 je určena pouze ke čtení
 :r3 je určena ke čtení i k zápisu
 :r4 se řadí mezi periferní paměti
:r3 ok
--
Doslovný překlad zkratky RAM je
 :r1 Rewrite And Machine
 :r2 Random Access Memory
 :r3 Record Access Memory
:r2 ok
--
Vestavěný program řídící činnost automatického jednoúčelového zařízení patří typicky do kategorie
 :r1 hardware
 :r2 bestware
 :r3 firmware
 :r4 adware
 :r5 spyware
:r3 ok
--
Jednotka informace 1 slovo (1 word) odpovídá
 :r1 80 b
 :r2 2 B
 :r3 32 b
 :r4 64 b
 :r5 všechny odpovědi mohou být správně
:r5 ok
--
Jedno slovo obyčejně <B>nemá</B>
 :r1 1 slabiku
 :r2 2 slabiky
 :r3 4 slabiky

 :r4 8 slabik
:r1 ok
--
Kontrolní bit například na děrné pásce se nazývá
 :r1 párový bit
 :r2 partikulární bit
 :r3 paralelní bit
 :r4 parciální bit
 :r5 paritní bit
:r5 ok
--
24bitová adresová sběrnice dokáže adresovat paměťový prostor o kapacitě maximálně (adresovatelná jednotka je bajt):
 :r1 4 MB
 :r2 16 MB
 :r3 1 GB
 :r4 4 GB
 :r5 16 GB
:r2 ok
--
Mezi různými typy pamětí nejmenší kapacitu má obvykle
 :r1 registr
 :r2 vnitřní (operační) paměť
 :r3 vnější (periferní) paměť
:r1 ok
--
Mezi různými typy pamětí je z hlediska přístupu nejrychlejší pamětí
 :r1 registr
 :r2 vnitřní (operační) paměť
 :r3 vnější (periferní) paměť
:r1 ok
--
Paměť se sekvenčním přístupem
 :r1 má vždy kratší přístupovou dobu k datům než paměť s přímým přístupem
 :r2 při přístupu k místu s adresou n projde nejdříve adresy 0-(n-1)
 :r3 je typicky paměť typu registr
 :r4 je typicky vnitřní (operační) paměť
:r2 ok
--
Která charakteristika <B>ne</B>platí pro paměť typu registr?
 :r1 velmi malá kapacita
 :r2 energeticky nezávislá
 :r3 velmi nízká přístupová doba
 :r4 paměť s přímým přístupem
 :r5 slouží pro krátkodobé uchování právě zpracovávaných informací
:r2 ok
--
Která charakteristika platí pro paměť typu registr?
 :r1 kapacita v řádu desítek GB
 :r2 energeticky nezávislá
 :r3 paměť s přímým přístupem
 :r4 slouží pro dlouhodobé uchování informací
 :r5 při přístupu k místu s adresou n projde nejdříve adresy 0-(n-1)
:r3 ok
--
Architektura počítače "von Neumann" obsahuje pravidlo:
 :r1 Počítač obsahuje procesor, DMA kanál, operační paměť a V/V zařízení.
 :r2 Počítač obsahuje operační paměť, ALJ, řadič a V/V zařízení.
 :r3 Počítač obsahuje procesor, DMA kanál a operační paměť.
:r2 ok
--

Architektura počítače "von Neumann" obsahuje pravidlo:
 :r1 Údaje a instrukce jsou vyjádřeny binárně.
 :r2 Údaje a instrukce jsou vyjádřeny číselně.
 :r3 Údaje a instrukce jsou vyjádřeny slovně.
 :r4 Instrukce se v assembleru píší zkratkou.
:r1 ok
--
V architektuře "von Neumann" má dekódování instrukcí na starost
 :r1 řadič
 :r2 aritmeticko-logická jednotka
 :r3 procesor
 :r4 operační paměť
 :r5 V/V zařízení
:r1 ok
--
Které tvrzení <B>ne</B>platí pro von Neumannovu architekturu?
 :r1 Program je uložen v paměti oddělené od paměti pro data.
 :r2 Počítač obsahuje operační paměť, ALJ, řadič a V/V zařízení.
 :r3 Program je uložen v paměti spolu s daty.
 :r4 Instrukce jsou vyjádřeny binárně.
 :r5 Data jsou vyjádřena binárně.
:r1 ok
--
Stavová hlášení jsou v architektuře "von Neumann" zasílána:
 :r1 aritmeticko-logické jednotce
 :r2 operační paměti
 :r3 řadiči
 :r4 V/V zařízení
 :r5 procesoru
:r3 ok
--
Které tvrzení o koncepci Johna von Neumanna <B>ne</B>platí?
 :r1 Program se umístí do operační paměti přes ALJ pomocí vstupního
zařízení.
 :r2 Data se umístí do operační paměti přes ALJ pomocí vstupního zařízení.
 :r3 Jednotlivé kroky výpočtu provádí aritmeticko-logická jednotka.
 :r4 Mezivýsledky jsou ukládány do operační paměti.
 :r5 Po skončení jsou výsledky poslány přes řadič na výstupní zařízení.
:r5 ok
--
Ve von Neumannově modelu
 :r1 netečou data z ALJ do paměti
 :r2 netečou data z řadiče do ALJ
 :r3 netečou data z ALJ do řadiče
 :r4 netečou data z paměti do ALJ
:r2 ok
--
Mezi typickou činnost řadiče patří
 :r1 transformuje instrukce na posloupnost signálů ovládající připojené
zařízení
 :r2 poskytuje paměťový prostor pro data, která tečou do procesoru
 :r3 slouží jako podpůrná výpočetní jednotka pro ALJ
 :r4 transformuje logickou adresu na fyzickou
:r1 ok
--
DMA je určeno především pro
 :r1 ukládání často užívaných instrukcí
 :r2 přenos dat z disku do operační paměti
 :r3 korekci obrazového výstupu
 :r4 kontrolu dat ukládaných na disk
 :r5 provádění aritmetických operací

:r2 ok
--
V polyadické soustavě je číslo
 :r1 součet bitů n-tice, ve které je uloženo.
 :r2 vždy dělitelné svým základem.
 :r3 součet mocnin základu vynásobených číslicemi.
:r3 ok
--
Čísla lze snadno (každou k-tici číslic nižší soustavy nahradíme číslicí soustavy vyšší) převádět mezi soustavami o základu
 :r1 5 a 7
 :r2 8 a 2
 :r3 10 a 16
:r2 ok
--
Číslo 21 v desítkové soustavě po převedení do soustavy dvojkové je
 :r1 10101
 :r2 11011
 :r3 10011
 :r4 nelze do dvojkové soustavy převést
:r1 ok
--
Pascalovský typ INTEGER je celé číslo, které se na počítačích PC zobrazuje v
 :r1 přímém kódu.
 :r2 doplňkovém kódu.
 :r3 inverzním kódu.
:r2 ok
--
Znaménkový bit v celém čísle je zpravidla bit
 :r1 nejnižšího řádu.
 :r2 nultého řádu.
 :r3 nejvyššího řádu.
:r3 ok
--
Znaménkový bit bývá zpravidla
 :r1 roven jedné, pokud se zobrazuje číslo kladné
 :r2 roven nule, pokud se zobrazuje číslo záporné
 :r3 roven nule, pokud se zobrazuje číslo kladné
:r3 ok
--
Rozsah zobrazení celého čísla uloženého ve dvojkovém doplňkovém kódu na 8 (celkem) bitech je
 :r1 <-128;127>
 :r2 <-256;255>
 :r3 <-511;512>
 :r4 <-1024;1023>
 :r5 žádný z uvedených
:r1 ok
--
Největší zobrazitelné celé číslo ve dvojkovém doplňkovém kódu má tvar
 :r1 100...00
 :r2 111...11
 :r3 000...00
 :r4 100...01
 :r5 011...11
:r5 ok
--
Při sčítání dvou čísel v inverzním kódu jako korekci výsledku použijeme:
 :r1 násobný přenos
 :r2 kruhový přenos

 :r3 konverzní přenos
 :r4 desítkový přenos
:r2 ok
--
Přeplnění (přetečení) je stav, ve kterém
 :r1 výsledek spadá mimo přesnost
 :r2 výsledek spadá mimo rozlišitelnost
 :r3 výsledek spadá mimo rozsah zobrazení
:r3 ok
--
Vyberte <B>ne</B>pravdivé tvrzení týkající se zobrazení celého čísla:
 :r1 přímý kód obsahuje kladnou a zápornou nulu
 :r2 inverzní kód obsahuje kladnou a zápornou nulu
 :r3 doplňkový kód obsahuje pouze jednu nulu
 :r4 rozsah zobrazení doplňkového kódu je symetrický
 :r5 se všemi bity doplňkového kódu se pracuje stejně
:r4 ok
--
Inverzní kód pro zobrazení celého čísla <B>ne</B>má
 :r1 jednu nulu
 :r2 symetrický rozsah zobrazení
 :r3 znaménkový bit
 :r4 ve znaménkovém bitu jedničku pro označení záporného čísla
:r1 ok
--
Znaménkový bit pro zobrazení celého čísla
 :r1 je bit nejnižšího řádu
 :r2 se běžně nepoužívá
 :r3 je bit nejnižšího řádu pouze pokud se jedná o číslo
 :r4 má hodnotu 1 pro kladné číslo
 :r5 má hodnotu 0 pro kladné číslo
:r5 ok
--
Přetečení v celočíselné aritmetice ve dvojkovém doplňkovém kódu nastane
 :r1 pokud se přenos ze znaménkového bitu rovná přenosu do znaménkového
bitu
 :r2 pokud se přenos ze znaménkového bitu nerovná přenosu do znaménkového
bitu
 :r3 pokud se přenos ze znaménkového bitu nerovná znaménkovému bitu
 :r4 pokud se přenos ze znaménkového bitu rovná znaménkovému bitu
 :r5 pokud výsledek operace nespadá mimo rozsah zobrazení
:r2 ok
--
Osmičkovou a šestnáctkovou soustavu používáme, protože:
 :r1 vnitřně si počítač uchovává data v těchto soustavách
 :r2 výpočet procesoru je rychlejší než při použití dvojkové soustavy
 :r3 zápis čísla je kratší než ve dvojkové soustavě
 :r4 vstupní a výstupní zařízení pracují s těmito soustavami
:r3 ok
--
Binární hodnota 0,1001 odpovídá dekadické hodnotě desetinného čísla:
 :r1 9/16
 :r2 1/32
 :r3 9/10
 :r4 1/16
 :r5 10/9
:r1 ok
--
Při sčítání ve dvojkovém doplňkovém kódu platí:
 :r1 přetečení nastane, pokud je rozsah zobrazení jiný než
<0;2ⁿ-1>

 :r2 všechny bity (kromě znaménkového) se sčítají stejně
 :r3 vznikne-li přenos ze znaménkového bitu, je nutné provádět tzv. kruhový přenos
 :r4 přetečení nastane, pokud se přenosy z/do znaménkového bitu rovnají
 :r5 vznikne-li přenos ze znaménkového bitu, tak se ignoruje
:r5 ok
--
Dvojkové číslo 1000 v přímém kódu v zobrazení se znaménkem na 4 bitech je:
 :r1 největší zobrazitelné
 :r2 nejmenší zobrazitelné
 :r3 kladná nula
 :r4 záporná nula
 :r5 žádná odpověď není správná
:r4 ok
--
Dvojkové číslo 1000 v inverzním kódu v zobrazení se znaménkem na 4 bitech je:
 :r1 největší zobrazitelné
 :r2 nejmenší zobrazitelné
 :r3 kladná nula
 :r4 záporná nula
 :r5 žádná odpověď není správná
:r2 ok
--
Dvojkové číslo 1111 v doplňkovém kódu v zobrazení se znaménkem na 4 bitech je:
 :r1 největší zobrazitelné
 :r2 nejmenší zobrazitelné
 :r3 kladná nula
 :r4 záporná nula
 :r5 žádná odpověď není správná
:r5 ok
--
Kruhový přenos je:
 :r1 inverze bitů
 :r2 inverze bitů a přičtení jedničky k výsledku
 :r3 přičtení přenosu z nejvyššího řádu k výsledku
 :r4 přičtení přenosu z nejvyššího řádu ke znaménkovému bitu
 :r5 přičtení jedničky k nejvyššímu řádu výsledku
:r3 ok
--
Kladná čísla v zobrazení se znaménkem mají na n bitech:
 :r1 ve všech kódech stejný rozsah
 :r2 v přímém kódu o 1 větší rozsah než v inverzním
 :r3 stejný rozsah jako kladná čísla v zobrazení bez znaménka
 :r4 v inverzním kódu o 1 číslo méně, než je záporných
 :r5 v inverzním kódu rozsah <0;2ⁿ-1>
:r1 ok
--
Které z dvojkových čísel v reprezentaci se znaménkem na 4 bitech je kladné?
 :r1 1010 v inverzním kódu
 :r2 0100 v inverzním kódu
 :r3 1010 v přímém kódu
 :r4 1111 v doplňkovém kódu
 :r5 všechny odpovědi jsou správné
:r2 ok
--
Kladná čísla v reprezentaci bez znaménka mají na n bitech rozsah:
 :r1 <0;2ⁿ-1>
 :r2 <0;2ⁿ⁻¹-1>
 :r3 <0;2ⁿ⁻¹+1>

 :r4 <-2ⁿ-1;2ⁿ-1>
 :r5 <-2ⁿ⁻¹-1;2ⁿ⁻¹-1>
:r1 ok
--
Rozsah zobrazení směrem ke kladným číslům a směrem k záporným číslům je rozložen asymetricky v:
 :r1 přímém kódu
 :r2 inverzním kódu
 :r3 doplňkovém kódu
 :r4 přímém a inverzním kódu
 :r5 inverzním a doplňkovém kódu
:r3 ok
--
Dvě reprezentace nuly se vyskytují v:
 :r1 přímém a doplňkovém kódu
 :r2 přímém a inverzním kódu
 :r3 inverzním a doplňkovém kódu
 :r4 doplňkovém, inverzním a přímém kódu
:r2 ok
--
Která z čísel jsou shodná (nejvyšší bit je znaménkový)?
 :r1 1001 v přímém a 1010 v inverzním kódu
 :r2 1101 v inverzním a 1110 v doplňkovém kódu
 :r3 1111 v doplňkovém a 1000 v přímém kódu
 :r4 1000 v doplňkovém a 1000 v inverzním kódu
 :r5 žádná z odpovědí není správná
:r2 ok
--
Rozsah zobrazení dvojkového doplňkového kódu na n bitech je:
 :r1 <0;2ⁿ⁻¹-1>
 :r2 <-2ⁿ⁻¹;2ⁿ⁻¹-1>
 :r3 <-2ⁿ⁻¹-1;2ⁿ⁻¹-1>
 :r4 <-2ⁿ-1;2ⁿ-1>
 :r5 <-2ⁿ+1;2ⁿ-1>
:r2 ok
--
Dvojkové číslo 1001 v reprezentaci se znaménkem na 4 bitech se v inverzním kódu rovná
 :r1 6
 :r2 -6
 :r3 9
 :r4 -9
:r2 ok
--
Jak při sčítání binárních čísel ve dvojkovém doplňkovém kódu poznám, že došlo k přetečení?
 :r1 k přetečení nemůže dojít, zabraňuje mu kruhový přenos
 :r2 přenos ze znaménkového bitu je 1
 :r3 přenos do znaménkového bitu se nerovná přenosu ze znaménkového bitu
 :r4 přenos do znaménkového bitu se rovná přenosu ze znaménkového bitu
:r3 ok
--
Číslo 14 v decimální soustavě odpovídá
 :r1 D v hexadecimální soustavě
 :r2 15 v oktalové soustavě
 :r3 1101 v binární soustavě
 :r4 E v hexadecimální soustavě
:r4 ok
--
Kruhový přenos v inverzním kódu se využívá
 :r1 pro korekci při přechodu přes nulu

 :r2 pro zkopírování nejnižšího bitu do nejvyššího
 :r3 pro zkopírování nejvyššího bitu do nejnižšího
 :r4 kruhový přenos se v inverzním kódu nepoužívá
:r1 ok
--
Jednoduše nelze převádět čísla mezi soustavami o základech
 :r1 5 a 25
 :r2 3 a 9
 :r3 4 a 40
 :r4 6 a 216
 :r5 6 a 36
:r3 ok
--
Osmičková soustava se také nazývá
 :r1 oktetová
 :r2 oktalová
 :r3 oktanová
 :r4 oktarová
 :r5 oklotová
:r2 ok
--
V ASCII kódu má
 :r1 ordinální hodnota znaku návrat vozíku (CR) menší hodnotu než ordinální hodnota znaku 'A'.
 :r2 ordinální hodnota znaku návrat vozíku (CR) větší hodnotu než ordinální hodnota znaku 'A'.
 :r3 znak návrat vozíku (CR) v ASCII kódu vůbec není.
:r1 ok
--
V ASCII kódu jsou znaky s ordinální hodnotou 0 až 31 označeny jako
 :r1 řídící znaky
 :r2 alfanumerické znaky
 :r3 alfabetické znaky
 :r4 tisknutelné znaky
:r1 ok
--
Písmena s diakritikou nejsou součástí vnějšího kódování
 :r1 ASCII
 :r2 ISO-8859-2
 :r3 Windows-1250
:r1 ok
--
Jaké kódování je korektní pro zobrazení všech českých znaků s diakritikou
 :r1 ASCII
 :r2 ISO-8859-1
 :r3 ISO-8859-2
:r3 ok
--
Znak "Line feed"
 :r1 je řídící znak s ordinální hodnotou nižší než 30
 :r2 je řídící znak s ordinální hodnotou vyšší než 30
 :r3 se nevyskytuje v kódování ASCII-7
 :r4 není řídící znak
:r1 ok
--
Řídící znak "Line feed" znamená
 :r1 přesun na začátek téhož řádku
 :r2 posun na další řádek se zachováním sloupce
 :r3 začátek příkazové řídicí sekvence
 :r4 přesun na začátek předchozího řádku
 :r5 takový řídící znak neexistuje

:r2 ok
--
Řídící znak "Carriage return" znamená
 :r1 přesun na začátek téhož řádku
 :r2 přesun na začátek dalšího řádku
 :r3 začátek příkazové řídicí sekvence
 :r4 přesun na začátek předchozího řádku
 :r5 takový řídící znak neexistuje
:r1 ok
--
Pro označení konce řádku v textovém souboru MS-Windows slouží kombinace znaků:
 :r1 CR+NUL
 :r2 CR+LF
 :r3 BS+CR
 :r4 LF
 :r5 CR+DEL
:r2 ok
--
Konec řádku v textovém souboru operačního systému UNIX se označuje kombinací řídicích znaků:
 :r1 CR+NUL
 :r2 CR+LF
 :r3 BS+CR
 :r4 LF
 :r5 CR+DEL
:r4 ok
--
Unicode je
 :r1 vnější kódování znaků
 :r2 sjednocené kódování celých čísel
 :r3 způsob ukládání reálných čísel
:r1 ok
--
UTF-8 zobrazuje jeden znak
 :r1 vždy jedním bajtem
 :r2 vždy dvěma bajty
 :r3 různým počtem bajtů
:r3 ok
--
Unicode je
 :r1 způsob uložení a UTF-8 je vnější kódování
 :r2 vnější kódování a UTF-8 je způsob uložení
:r2 ok
--
Česká písmena s diakritikou jsou v UTF-8 uložena nejvíce na
 :r1 jednom bajtu
 :r2 dvou bajtech
 :r3 třech bajtech
 :r4 čtyřech bajtech
:r2 ok
--
UTF-8 uloží znak z ASCII 7 na
 :r1 1 bajtu
 :r2 2 bajty
 :r3 3 bajty
 :r4 4 bajty
 :r5 5 bajtů
:r1 ok
--

Počet bajtů, v kolika je uložen znak v UTF-8 (je-li uložen ve více než jednom bajtu), je vyjádřen
 :r1 počtem binárních jedniček v bitech nejvyšších řádů
 :r2 počtem binárních nul v bitech nejvyšších řádů
 :r3 číslem 0-7 v nejvyšších třech bitech
 :r4 číslem 0-7 v nejnižších třech bitech
:r1 ok
--
Kolik bitů z UNICODE kódování lze zobrazit pomocí původní specifikace UTF-8, která povolovala jeden znak zobrazovat až na 6 bajtech?
 :r1 31
 :r2 48
 :r3 15
 :r4 16
 :r5 32
:r1 ok
--
Zobrazíme-li znak z kódování ASCII-7 v kódování UTF-8, co bude v bitu nejvyššího řádu?
 :r1 vždy 0
 :r2 vždy 1
 :r3 0 nebo 1 podle znaku
:r1 ok
--
Vnější kódy ISO-8859-2 a Windows-1250 se liší v ordinální hodnotě znaku
 :r1 ň
 :r2 č
 :r3 š
:r3 ok
--
Ve kterém vnějším kódování <B>ne</B>jsou česká písmena s diakritikou?
 :r1 ASCII
 :r2 ISO-8859-2
 :r3 Windows-1250
 :r4 UNICODE
:r1 ok
--
Detekční kód je kód, který
 :r1 nahlásí chybu v počítači.
 :r2 rozpozná chybu v uložené či přenášené informaci.
 :r3 detekuje hackera v počítači.
:r2 ok
--
Opravný kód je kód, který
 :r1 najde chybu v systému Windows a opraví ji.
 :r2 opraví chybu programátora v jeho zdrojovém kódu.
 :r3 opraví chybu v uložené či přenášené informaci.
:r3 ok
--
Hammingova trojrozměrná krychle má
 :r1 6 stěn.
 :r2 2 stěny.
 :r3 žádnou stěnu.
 :r4 8 stěn.
:r1 ok
--
BCD (Binary Coded Decimal) znamená
 :r1 binárně zakódovaná čísla tak, aby je nešlo dešifrovat.
 :r2 desítkově kódovaná binární čísla.
 :r3 jedna desítková číslice uložená vždy na čtyřech bitech.
:r3 ok

--
BCD znamená
 :r1 Binary Coded Decimal
 :r2 Binary Crowded Decimal
 :r3 Binary Coded Hexadecimal
 :r4 Bipolary Coded Decimal
:r1 ok
--
BCD kód v každé
 :r1 trojici bitů ukládá jednu oktalovou číslici
 :r2 čtveřici bitů ukládá jednu šestnáctkovou číslici
 :r3 čtveřici bitů ukládá jednu desítkovou číslici
 :r4 trojici bitů ukládá jednu desítkovou číslici
:r3 ok
--
Kladné číslo v rozvinutém BCD tvaru je
 :r1 71346C
 :r2 71346D
 :r3 F7F1F3F4C6
 :r4 F7F1F3F4F6C
 :r5 +F7F1F3F4F6D
:r3 ok
--
Číslo, které je v rozvinutém BCD tvaru uloženo na 5 bajtech, bude ve zhuštěném BCD tvaru uloženo ve
 :r1 2 bajtech
 :r2 3 bajtech
 :r3 4 bajtech
 :r4 5 bajtech
 :r5 6 bajtech
:r2 ok
--
V čem je uznávaná výhoda zobrazení čísel v BCD kódu oproti zobrazení čísel v přímém binárním kódu?
 :r1 jednodušší převod čísla do desítkové soustavy
 :r2 jednodušší provádění aritmetických operací
 :r3 kratší zápis čísla
 :r4 BCD kód je nyní všeobecně používanější
:r1 ok
--
Co znamená kód 2 z 5?
 :r1 způsob zabezpečení informace, právě dva bity jsou rovny nule
 :r2 způsob kódování podobný kódu CP1250
 :r3 způsob zabezpečení, právě dva bity jsou rovny jedné
 :r4 způsob kódování na principu UTF-16
:r3 ok
--
Při Hammingově vzdálenosti (d) pět
 :r1 mohu kód opravit, pokud vznikne maximálně jedna chyba
 :r2 mohu kód opravit, pokud vzniknou maximálně dvě chyby
 :r3 mohu kód opravit, pokud vzniknou maximálně tři chyby
 :r4 nejsem schopen opravit chybu
:r2 ok
--
Při Hammingově vzdálenosti (d) dva
 :r1 jsem schopen detekovat chybu a nejsem schopen ji opravit
 :r2 jsem schopen detekovat chybu a jsem schopen ji opravit
 :r3 nejsem schopen detekovat chybu
:r1 ok
--
Sudá parita znamená

```
 :r1 počet bitů vč. paritního obsahujících hodnotu 1 je sudý
 :r2 počet bitů vč. paritního obsahujících hodnotu 1 je lichý
 :r3 počet bitů bez paritního obsahujících hodnotu 1 je sudý
 :r4 počet bitů bez paritního obsahujících hodnotu 1 je lichý
 :r5 počet chyb, které jsme schopni detekovat, je sudý
:r1 ok
--
Mějme detekční kód 2 z 5. Které z následujících čísel obsahuje chybu?
 :r1 00101
 :r2 11010
 :r3 10001
 :r4 00011
 :r5 01100
:r2 ok
--
Ztrojení
 :r1 je příkladem vnějšího kódu
 :r2 je příkladem opravného kódu
 :r3 uloží hodnotu tří bitů na jeden bit
 :r4 umožňuje detekovat 3 chyby, ale pouze 2 opravit
:r2 ok
--
Kódová (Hammingova) vzdálenost je:
 :r1 počet bitů, v nichž se liší dvě sousední platné kódové kombinace
 :r2 počet bitů, v nichž se se shodují dvě sousední platné kódové kombinace
 :r3 počet jedničkových bitů ve dvou sousedních platných kódových
kombinacích
 :r4 počet chyb, které jsme schopni detekovat
 :r5 počet chyb, které jsme schopni opravit
:r1 ok
--
Pro Hammingovu vzálenost 1 platí
 :r1 žádnou chybu nelze detekovat, tedy ani opravit
 :r2 jednu chybu lze detekovat, ale nelze ji opravit
 :r3 jednu chybu lze detekovat a je možné ji opravit
 :r4 dvě chyby lze detekovat a jednu chybu lze opravit
:r1 ok
--
Kolik chyb jsme schopni detekovat, jestliže kódová vzdálenost d=3?
 :r1 žádnou
 :r2 jednu
 :r3 dvě
 :r4 tři
 :r5 čtyři
:r3 ok
--
Kolik chyb jsme schopni opravit, jestliže kódová vzdálenost d=3?
 :r1 žádnou
 :r2 jednu
 :r3 dvě
 :r4 tři
 :r5 čtyři
:r2 ok
--
V opravném kódu v případě ztrojení každého bitu
 :r1 jsme schopni jednu chybu detekovat a dvě chyby korektně opravit
 :r2 jsme schopni jednu chybu detekovat a jednu chybu korektně opravit
 :r3 jsme schopni dvě chyby detekovat a obě dvě korektně opravit
 :r4 jsme schopni dvě chyby detekovat a jednu chybu korektně opravit
:r4 ok
--
```

Jaké ordinální hodnoty mají číslice v EBCDIC (vnější kód BCD)?
 :r1 A0 až A9
 :r2 C0 až C9
 :r3 D0 až D9
 :r4 E0 až E9
 :r5 F0 až F9
:r5 ok
--
Co znamená Big-Endian
 :r1 počítač má jeden konec větší než druhý
 :r2 bajt nejvyššího řádu je na nejnižší adrese
 :r3 bajt nejnižšího řádu je na nejnižší adrese
 :r4 bajt nejvyššího řádu je na nejvyšší adrese
:r2 ok
--
Co znamená použití pořadí Little-Endian?
 :r1 Bajt nejnižšího řádu je uložen na nejnižší adrese.
 :r2 Bajt nejvyššího řádu je uložen na nejnižší adrese.
 :r3 Bajt nejnižšího řádu je uložen na nejvyšší adrese.
 :r4 Všechny bity (kromě znaménkového) se sčítají stejně.
:r1 ok
--
Little-Endian a Big-Endian jsou způsoby
 :r1 ukládání bitů v bajtu
 :r2 ukládání bajtů ve slově
 :r3 připojování konektorů sběrnic
:r2 ok
--
Jak na čísle ve dvojkovém doplňkovém kódu poznáme, zda je uloženo v Big-Endian nebo Little-Endian
 :r1 podle hodnoty nejvyššího bitu
 :r2 podle hodnoty nejvyššího bajtu
 :r3 podle hodnoty nejnižšího bajtu
 :r4 podle hodnoty nejnižšího bitu
 :r5 nelze to ze zápisu čísla jednoznačně poznat
:r5 ok
--
Mezi operace Booleovy algebry <B>ne</B>patří
 :r1 logický součet
 :r2 logický rozdíl
 :r3 logický součin
 :r4 negace
:r2 ok
--
Sériové zapojení vyjádřené v Booleově algebře znamená
 :r1 logický součet
 :r2 logický rozdíl
 :r3 logický součin
 :r4 negaci
:r3 ok
--
Paralelní zapojení vyjádřené v Booleově algebře znamená
 :r1 logický součet
 :r2 logický rozdíl
 :r3 logický součin
 :r4 negaci
:r1 ok
--
Který z uvedených způsobů se nepoužívá pro minimalizaci výrazu?
 :r1 matematické úpravy
 :r2 jednotková krychle

 :r3 karnaughova mapa
 :r4 jednotková kružnice
:r4 ok
--
Proč není Booleova algebra vhodná pro technickou realizaci?
 :r1 obsahuje příliš mnoho operací
 :r2 byla vymyšlena dříve, než se začala uplatňovat von Neumannova koncepce
 :r3 zakreslení grafů je pomocí ní příliš obtížné
 :r4 není možné pomocí ní provádět operaci implikace
:r1 ok
--
Jaké operace využívá Shefferova algebra?
 :r1 jedinou operaci a to negovaný logický součin (NAND)
 :r2 jedinou operaci a to negovaný logický součet (NOR)
 :r3 dvě operace - negovaný logický součin (NAND) a negovaný logický součet (NOR)
 :r4 operace logický součin (AND), logický součet (OR) a negaci (NOT)
:r1 ok
--
Shefferova algebra (NAND) se používá místo Booleovy algebry v technických zapojeních, protože
 :r1 je rychlejší.
 :r2 je levnější.
 :r3 má jen jednu operaci.
 :r4 má více operací.
:r3 ok
--
Zakázané pásmo v obvodech
 :r1 je vymezeno nejnižší hodnotou napětí, při které již může dojít k poškození obvodu
 :r2 vymezuje hodnoty signálu, ve kterých se signál nesmí nacházet během jeho vzorkování
 :r3 je maximální vzdálenost mezi dvěma obvody, ve které ještě dochází k nežádoucímu ovlivňování tvaru signálu
:r2 ok
--
Zakázané pásmo v obvodech je
 :r1 vzdálenost od počítače, ve které se nesmí vyskytovat jiný spotřebič.
 :r2 poloměr kruhu okolo procesoru, ve kterém se nesmí vyskytovat žádný signál.
 :r3 rozsah hodnot, ve kterém se signál nesmí nacházet v okamžiku vzorkování.
:r3 ok
--
Napájecí napětí technologie TTL je
 :r1 5 V
 :r2 220 V
 :r3 120 V na americkém kontinentu
:r1 ok
--
Invertor
 :r1 je sekvenční logický člen
 :r2 je logický člen měnící kladné napětí na záporné
 :r3 je logický člen měnící logickou 0 na logickou 1 a opačně
 :r4 je sekvеční logický člen měnící logickou 0 na logickou 1 a opačně
:r3 ok
--
Výstupní hodnota logického členu NOR je rovna 1, když
 :r1 všechny vstupní hodnoty jsou 1.
 :r2 aspoň jedna vstupní hodnota je 0.
 :r3 aspoň jedna vstupní hodnota je 1.

 :r4 všechny vstupní hodnoty jsou 0.
:r4 ok
--
Výstupní hodnota logického členu NOR je rovna 0, když
 :r1 aspoň jedna vstupní hodnota je 0.
 :r2 aspoň jedna vstupní hodnota je 1.
 :r3 všechny vstupní hodnoty jsou 0.
:r2 ok
--
Výstupní hodnota logického členu NAND je rovna 0, když
 :r1 všechny vstupní hodnoty jsou 1.
 :r2 aspoň jedna vstupní hodnota je 0.
 :r3 aspoň jedna vstupní hodnota je 1.
 :r4 všechny vstupní hodnoty jsou 0.
:r1 ok
--
Výstupní hodnota logického členu NAND je rovna 1, když
 :r1 všechny vstupní hodnoty jsou 1.
 :r2 aspoň jedna vstupní hodnota je 0.
 :r3 aspoň jedna vstupní hodnota je 1.
:r2 ok
--
Mezi kombinační logické obvody patří
 :r1 NAND, NOT, multiplexor
 :r2 RS, JK, AND, OR
 :r3 NOR, D, XOR
:r1 ok
--
Mezi kombinační logické obvody patří
 :r1 klopný obvod R-S
 :r2 sčítačka pro jeden binární řád
 :r3 jednobitová paměť
:r2 ok
--
Kombinační logický obvod "nonekvivalence" má stejnou funkci jako:
 :r1 logický součet
 :r2 sčítačka modulo 2
 :r3 multiplexor
:r2 ok
--
Klopný obvod RS v obecném případě nesmí mít na vstupu kombinaci 00,
 :r1 pokud je řízen jedničkami
 :r2 pokud je řízen nulami
 :r3 protože na komplementárních výstupech budou stejné hodnoty
:r2 ok
--
Parita je
 :r1 obvod pro vyhodnocení hlasovací funkce.
 :r2 způsob porovnání dvou čísel.
 :r3 způsob zabezpečení informace proti chybě.
:r3 ok
--
Multiplexor se čtyřmi datovými vstupy je obvod, který
 :r1 dle zadané adresy vybere jeden ze vstupních signálů a předá jej na
výstup.
 :r2 dle zadané adresy vybere čtyři vstupní signály a sloučí je do jednoho
výstupního.
 :r3 vybere náhodně jeden ze čtyř vstupních signálů a předá jej na výstup.
:r1 ok
--
Multiplexor se 16 datovými vstupy potřebuje

 :r1 4 adresové vstupy.
 :r2 16 adresových vstupů.
 :r3 65536 adresových vstupů.
:r1 ok
--
Multiplexor, který má celkem 6 vstupů, má
 :r1 2 datové a 4 adresové vstupy
 :r2 3 datové a 3 adresové vstupy
 :r3 4 datové a 2 adresové vstupy
:r3 ok
--
Dekodér, který má 2 vstupy, má
 :r1 2 výstupy.
 :r2 4 výstupy.
 :r3 8 výstupů.
:r2 ok
--
Dekodér, který má 8 vstupů, má
 :r1 2 výstupy
 :r2 3 výstupy
 :r3 8 výstupů
 :r4 64 výstupů
 :r5 256 výstupů
:r5 ok
--
Dekodér, který má 4 vstupy, má
 :r1 2 výstupy
 :r2 3 výstupy
 :r3 4 výstupy
 :r4 8 výstupů
 :r5 16 výstupů
:r5 ok
--
Úplná sčítačka pro jeden binární řád má
 :r1 dva bity sčítanců na vstupu a jeden bit součtu na výstupu.
 :r2 dva bity sčítanců na vstupu a jeden bit součtu a přenos na výstupu.
 :r3 dva bity sčítanců a přenos na vstupu a jeden bit součtu a přenos na
výstupu.
:r3 ok
--
Co je pravda?
 :r1 Sekvenční logické obvody mají vnitřní stav.
 :r2 Kombinační logické obvody mají vnitřní stav.
 :r3 Nic z toho není pravda.
:r1 ok
--
Zakázaný stav u klopného obvodu R-S řízeného jedničkami je stav, kdy
 :r1 R=0 a S=0.
 :r2 R=1 a S=1.
 :r3 se R a S nerovnají.
 :r4 je R nebo S nenastaveno.
:r2 ok
--
Klopný obvod je název obvodu
 :r1 ze skupiny sčítaček.
 :r2 ze skupiny kombinačních logických obvodů.
 :r3 ze skupiny sekvenčních logických obvodů.
:r3 ok
--
Sčítačka pro jeden řád BCD kódu se realizuje pomocí dvou čtyřbitových
sčítaček. Pokud je součet dvou BCD číslic klasickou sčítačkou větší než 9

 :r1 provádí se korekce přičtením čísla 6.
 :r2 provádí se korekce extrakcí dolních 4 bitů.
 :r3 není třeba dělat korekci, přenos se použije jako číslice vyššího řádu.
:r1 ok
--
Žádný bit se neztrácí při
 :r1 logickém posunu bitů.
 :r2 rotaci bitů.
 :r3 aritmetickém posunu doleva.
:r2 ok
--
Aritmetickým posunem doleva se
 :r1 násobí
 :r2 dělí
:r1 ok
--
Násobení dvěma lze realizovat
 :r1 rotací o jeden bit doprava.
 :r2 aritmetickým posunem o jeden bit doprava.
 :r3 aritmetickým posunem o jeden bit doleva.
:r3 ok
--
Operaci celočíselného dělení dvěma lze provést
 :r1 aritmetickým posuvem obsahu registru doleva
 :r2 logický posuvem obsahu registru doleva
 :r3 logický posuvem obsahu registru doprava
 :r4 aritmetickým posuvem obsahu registru doprava
:r4 ok
--
Aritmetickým posunem doprava dojde u kladného čísla typicky ke
 :r1 zvětšení čísla
 :r2 zmenšení čísla
:r2 ok
--
Co není správně?
 :r1 Boolova algebra je nauka o operacích na dvouprvkové množině
 :r2 Boolova algebra užívá tři základní operace
 :r3 Boolova algebra je vybudována na operaci negovaného logického součinu
:r3 ok
--
Technologie TTL používá jako svůj základní prvek
 :r1 tranzistor NPN
 :r2 tranzistor PNP
 :r3 invertor
 :r4 magnetické obvody
:r1 ok
--
Pro technickou realizaci je nejméně vhodná
 :r1 Booleova algebra
 :r2 Pierceova algebra
 :r3 Shefferova algebra
 :r4 všechny algebry jsou stejně vhodné
:r1 ok
--
Shefferova algebra je vybudována pouze na jediné logické operaci, a to
 :r1 NAND
 :r2 NOR
 :r3 XOR
 :r4 NOXOR
 :r5 AND
:r1 ok

--
Piercova algebra je vybudována pouze na jediné logické operaci, a to
 :r1 NAND
 :r2 NOR
 :r3 XOR
 :r4 NOXOR
 :r5 OR
:r2 ok
--
Základním stavebním prvkem technologie TTL je
 :r1 relé
 :r2 elektronka
 :r3 unipolární tranzistor
 :r4 bipolární tranzistor
:r4 ok
--
Logický obvod NAND
 :r1 pro vstupy 0 a 0 dá výstup 0
 :r2 pro vstupy 0 a 0 dá výstup 1
 :r3 pro vstupy 0 a 1 dá výstup 0
 :r4 pro vstupy 1 a 1 dá výstup 1
 :r5 provádí negaci logického součtu
:r2 ok
--
Logický obvod NOR
 :r1 pro vstupy 0 a 0 dá výstup 0
 :r2 pro vstupy 0 a 1 dá výstup 1
 :r3 pro vstupy 1 a 0 dá výstup 0
 :r4 pro vstupy 1 a 1 dá výstup 1
 :r5 provádí negaci logického součinu
:r3 ok
--
Logický obvod XOR (nonekvivalence)
 :r1 pro vstupy 0 a 0 dá na výstup 0
 :r2 pro vstupy 0 a 1 dá na výstup 0
 :r3 pro vstupy 1 a 1 dá na výstup 1
 :r4 pro vstupy 0 a 0 dá na výstup 1
 :r5 provádí negaci vstupu
:r1 ok
--
Negaci bitu provádí:
 :r1 logický obvod AND
 :r2 logický obvod OR
 :r3 invertor
 :r4 multiplexor
 :r5 dekodér
:r3 ok
--
Pro výběr jednoho z n vstupů slouží:
 :r1 logický obvod AND
 :r2 logický obvod NOR
 :r3 invertor
 :r4 multiplexor
 :r5 dekodér
:r4 ok
--
n adresových vstupů a 2ⁿ datových výstupů má:
 :r1 logický obvod AND
 :r2 logický obvod NOR
 :r3 invertor
 :r4 multiplexor

```
 :r5 dekodér
:r5 ok
--
Impuls je
 :r1 trvalá změna hodnoty signálu
 :r2 dočasná změna hodnoty signálu
 :r3 invertování hodnoty bitu
:r2 ok
--
Mezi sekvenční logické obvody patří
 :r1 multiplexor, dekodér, sčítačka modulo 2
 :r2 polosčítačka, klopný obvod JK, klopný obvod RS
 :r3 klopný obvod JK, klopný obvod RS, klopný obvod D
 :r4 žádná z uvedených možností
:r3 ok
--
Zakázaný stav se nachází u
 :r1 u polosčítačky
 :r2 klopného obvodu D
 :r3 klopného obvodu JK
 :r4 žádná z uvedených možností
:r4 ok
--
Sekvenční logické obvody se vyznačují tím, že
 :r1 výstup nezávisí na předchozí posloupnosti změn
 :r2 nemají vnitřní paměť
 :r3 výstup závisí na předchozí posloupnosti změn
 :r4 nemají tvz. zpětnou vazbu
:r3 ok
--
Výstupy z eventuální sčítačky Modulo 4 mohou nabývat hodnoty
 :r1 0, 1
 :r2 0, 1, 2
 :r3 0, 1, 2, 3
 :r4 0, 1, 2, 3, 4
:r3 ok
--
Pro kombinační logické obvody platí, že
 :r1 nepatří sem sčítačka modulo 2
 :r2 výstupy nezávisí na předchozí posloupnosti změn
 :r3 patří sem klopný obvod RS
 :r4 výstupy závisí na předchozí posloupnosti změn
:r2 ok
--
Signálem Reset
 :r1 je návrat do předem definovaného stavu
 :r2 není návrat do předem definovaného stavu
 :r3 vynulujeme všechny výstupní hodnoty
 :r4 všem vstupním hodnotám přiřadíme jedničku
:r1 ok
--
Mezi kombinační logické obvody <B>ne</B>patří
 :r1 polosčítačka
 :r2 multiplexor
 :r3 sčítačka modulo 2
 :r4 žádná z uvedených možností
:r4 ok
--
Zakázaný stav klopného obvodu JK nastane když
 :r1 J=0, K=0
 :r2 J=1, K=1
```

 :r3 J=1, K=0
 :r4 žádná z uvedených možností
:r4 ok
--
Korekce pro BCD sčítačku <B>ne</B>přičítá šestku, když
 :r1 bity součtu binárního řádu 1 a 3 jsou rovny jedné
 :r2 bity součtu binárního řádu 2 a 3 jsou rovny jedné
 :r3 přenosový bit součtu je roven jedné
 :r4 přenosový bit součtu je roven nule
:r4 ok
--
Logický posun nenulového obsahu registru doprava
 :r1 nikdy neovlivní znaménko
 :r2 nejvyššímu bitu přiřadí jedničku
 :r3 nejnižší bit se ztrácí
 :r4 žádná z uvedených možností
:r3 ok
--
Aritmetický posun nenulového obsahu registru doleva způsobí
 :r1 obsah registru se celočíselně vydělí dvěma, nezmění se znaménko,
nedošlo-li k přetečení
 :r2 obsah registru se celočíselně vynásobí dvěma, nezmění se znaménko,
nedošlo-li k přetečení
 :r3 obsah registru ani znaménko se nezmění
 :r4 obsah registru i znaménko se změní, pokud nedošlo k přetečení
:r2 ok
--
Pokud se obsah registru posune aritmeticky doprava a číslo se blíží k
maximální hodnotě, kterou lze do registru uložit, pak
 :r1 obsah bude celočíselně vydělen dvěma
 :r2 obsah bude vynásoben dvěma a výsledek bude správný
 :r3 obsah registru přeteče
 :r4 žádná z uvedených možností
:r1 ok
--
Jednotka Baud udává
 :r1 počet bajtů přenesených za sekundu
 :r2 počet bitů přenesených za sekundu
 :r3 počet změn stavů přenesených za sekundu
:r3 ok
--
Při stejné přenosové rychlosti je vždy počet bitů přenesených za sekundu
 :r1 menší nebo roven počtu baudů
 :r2 větší nebo roven počtu baudů
 :r3 menší než počet baudů
 :r4 větší než počet baudů
 :r5 rovný počtu baudů
:r2 ok
--
Jako sčítačka modulo 2, která neřeší přenosy, funguje
 :r1 logický člen NOR
 :r2 logický člen NAND
 :r3 logický člen XOR
 :r4 klopný obvod D
 :r5 klopný obvod RS
:r3 ok
--
Polosčítačka se dvěma vstupy
 :r1 má tři výstupy
 :r2 řeší přenos z nižšího řádu
 :r3 její pravdivostní tabulka má 8 řádků

 :r4 dává na výstup přenos do vyššího řádu
:r4 ok
--
Klopný obvod RS řízený nulami
 :r1 nemá zakázaný stav
 :r2 nemá definovaný stav pro vstupy 1 a 1
 :r3 pro hodnoty 1 a 1 setrvává v předchozím stavu
 :r4 pro hodnoty 0 a 0 setrvává v předchozím stavu
:r3 ok
--
"R" v názvu klopného obvodu RS znamená
 :r1 repeat
 :r2 reset
 :r3 read
 :r4 random
 :r5 ready
:r2 ok
--
Registry jsou typicky konstruovány z
 :r1 klopného obvodu D
 :r2 klopného obvodu JK
 :r3 klopného obvodu RS
 :r4 polosčítačky
 :r5 úplné sčítačky
:r1 ok
--
Při dvoustavové komunikaci je rychlost přenosu udávaná v baudech (Bd)
 :r1 větší než rychlost udávaná v bitech za sekundu
 :r2 menší než rychlost udávaná v bitech za sekundu
 :r3 stejná jako rychlost udávaná v bitech za sekundu
 :r4 neporovnatelná s rychlostí udávanou v bitech za sekundu
:r3 ok
--
Při čtyřstavové komunikaci je rychlost přenosu udávaná v baudech (Bd)
 :r1 větší než rychlost udávaná v bitech za sekundu
 :r2 menší než rychlost udávaná v bitech za sekundu
 :r3 stejná jako rychlost udávaná v bitech za sekundu
 :r4 neporovnatelná s rychlostí udávanou v bitech za sekundu
:r2 ok
--
Pod pojmem "zakázané pásmo" při přenosu signálu rozumíme
 :r1 skupinu počítačů, ke kterým signál nesmí dorazit
 :r2 frekvenci, se kterou nesmí vysílající vysílat
 :r3 rozsah napětí, v jehož rámci je hodnota signálu nedefinovaná
 :r4 všechny hodnoty napětí nerovnající se U_l a U_h
:r3 ok
--
Pro multiplexor <B>ne</B>platí
 :r1 má datové vstupy
 :r2 má adresové vstupy
 :r3 má datový výstup
 :r4 má adresový výstup
:r4 ok
--
Jaký zakázaný stav má klopný obvod RS řízený jedničkami?
 :r1 0,0
 :r2 0,1
 :r3 1,0
 :r4 1,1
:r4 ok
--

Pod rotací bitů vlevo rozumíme
 :r1 posuv z nižšího řádu do vyššího, žádná hodnota bitu se neztrácí
 :r2 posuv z nižšího řádu do vyššího, ztrácí se hodnota některého bitu
 :r3 posuv z vyššího řádu do nižšího, žádná hodnota bitu se neztrácí
 :r4 posuv z vyššího řádu do nižšího, ztrácí se hodnota některého bitu
:r1 ok
--
Pod rotací bitů vpravo rozumíme
 :r1 posuv z nižšího řádu do vyššího, žádná hodnota bitu se neztrácí
 :r2 posuv z nižšího řádu do vyššího, ztrácí se hodnota některého bitu
 :r3 posuv z vyššího řádu do nižšího, žádná hodnota bitu se neztrácí
 :r4 posuv z vyššího řádu do nižšího, ztrácí se hodnota některého bitu
:r3 ok
--
Pod pojmem logický posun vlevo rozumíme
 :r1 posuv z nižšího řádu do vyššího, žádná hodnota bitu se neztrácí
 :r2 posuv z nižšího řádu do vyššího, ztrácí se hodnota některého bitu
 :r3 posuv z vyššího řádu do nižšího, žádná hodnota bitu se neztrácí
 :r4 posuv z vyššího řádu do nižšího, ztrácí se hodnota některého bitu
:r2 ok
--
Pod pojmem logický posun vpravo rozumíme
 :r1 posuv z nižšího řádu do vyššího, žádná hodnota bitu se neztrácí
 :r2 posuv z nižšího řádu do vyššího, ztrácí se hodnota některého bitu
 :r3 posuv z vyššího řádu do nižšího, žádná hodnota bitu se neztrácí
 :r4 posuv z vyššího řádu do nižšího, ztrácí se hodnota některého bitu
:r4 ok
--
Při aritmetickém posunu
 :r1 se mění hodnota znaménkového bitu, nedojde-li k přetečení
 :r2 se nemění hodnota znaménkového bitu, nedojde-li k přetečení
 :r3 je posun doleva ekvivalentní celočíselnému dělení dvěma
 :r4 je posun doprava ekvivalentní násobení dvěma
:r2 ok
--
V techologii TTL při použití tranzistoru NPN se kolektor a emitor otevírá
 :r1 když je na bázi přivedena vysoká úroveň -- logická jednička
 :r2 když je na bázi přivedena nízká úroveň -- logická nula
 :r3 když je na kolektor přivedena vysoká úroveň -- logická jednička
 :r4 když je na kolektor přivedena nízká úroveň -- logická nula
:r1 ok
--
K čemu se využívá Karnaughova mapa
 :r1 k minimalizaci počtu operací B-algebry
 :r2 k uchování informace o rámcích, které nejsou zaplněny
 :r3 k uchování informace o dostupných V/V branách
 :r4 pro popis volných bloků paměti
:r1 ok
--
Pokud jsou 1 a 1 na vstupu sčítačky modulo 2, pak na výstupu je
 :r1 0
 :r2 1
 :r3 2
 :r4 tento vstup je neplatný
:r1 ok
--
Mám 16 zařízení, zařízení číslo 10 chci poslat signál 1, ostatním 0. Co použiji?
 :r1 dekodér
 :r2 multiplexor
 :r3 úplnou sčítačku

 :r4 polosčítačku
:r1 ok
--
Pro úplnou sčítačku pro jeden binární řád platí
 :r1 má 3 vstupy a 2 výstupy
 :r2 má 2 vstupy a 3 výstupy
 :r3 má 2 vstupy a 2 výstupy
 :r4 má 3 vstupy a 3 výstupy
:r1 ok
--
Co platí pro klopný obvod D?
 :r1 je to paměť na jeden bit
 :r2 má čtyři výstupy
 :r3 má čtyři datové vstupy
 :r4 má ekvivalentní funkci jako polosčítačka
:r1 ok
--
NOXOR je stejný jako:
 :r1 ekvivalence
 :r2 NOR
 :r3 OR
 :r4 NAND
:r1 ok
--
Které zapojení nelze popsat pomocí Booleovy algebry?
 :r1 sériové
 :r2 můstkové
 :r3 paralelní
 :r4 sérioparalelní
:r2 ok
--
Která paměť musí být energeticky nezávislá?
 :r1 vnější paměť
 :r2 vnitřní paměť
 :r3 registry
:r1 ok
--
Obsah adresového registru paměti se na výběr jednoho z výběrových
(adresových) vodičů převádí
 :r1 multiplexorem 1 z N.
 :r2 dekodérem 1 z N.
 :r3 sčítačkou 1 plus N.
:r2 ok
--
K destruktivnímu nevratnému zápisu do permanentní paměti pomocí přepalování
tavných spojek proudovými impulsy je určena paměť
 :r1 ROM
 :r2 PROM
 :r3 EPROM
:r2 ok
--
Parametr pamětí "vybavovací doba - čas přístupu" bude nejvyšší u
 :r1 registru
 :r2 vyrovnávací (cache) paměti
 :r3 operační paměti
 :r4 diskové paměti
:r4 ok
--
Paměť, která svůj obsah adresuje klíčem, který je uložen odděleně od obsahu
paměti a vyhledává se v klíči paralelně, se nazývá
 :r1 operační paměť.

 :r2 permanentní paměť.
 :r3 asociativní paměť.
 :r4 klíčová paměť.
:r3 ok
--
Paměť typu cache nebývá umístěna mezi
 :r1 procesorem a pamětí
 :r2 procesorem a V/V zařízením
 :r3 procesorem a registry
:r3 ok
--
Do paměti typu PROM
 :r1 nelze data zapsat
 :r2 lze zapsat data pouze jednou
 :r3 lze zapsat data libovolněkrát působením UV záření
 :r4 lze zapsat data libovolněkrát vyšší hodnotou elektrického proudu
 :r5 lze zapsat data libovolněkrát přepálením tavné pojistky NiCr
:r2 ok
--
Které tvrzení <B>ne</B>platí pro popis fyzické struktury vnitřní paměti?
 :r1 Dekodér na jeden z adresových vodičů nastaví hodnotu logická 1.
 :r2 Informace je na koncích datových vodičů zesílena zesilovačem.
 :r3 Adresa je přivedena na vstup dekodéru.
 :r4 Podle zapojení buněk na řádku projde/neprojde logická 1 na datové
vodiče.
 :r5 Datový registr má na vstup přivedeny adresové vodiče.
:r5 ok
--
Máme-li vnitřní paměť o kapacitě 16 bitů zapojenou jako matici paměťových
buněk 4x4 bity, pak nejmenší adresovatelná jednotka je
 :r1 1 bit
 :r2 2 bity
 :r3 4 bity
 :r4 16 bitů
 :r5 65536 bitů
:r3 ok
--
Působením UV záření je možné vymazat obsah paměti
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 EEPROM
 :r5 RAM
:r3 ok
--
Statickou, energeticky nezávislou pamětí není paměť typu
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 EEPROM
 :r5 žádná z odpovědí není správně
:r5 ok
--
Vybavovací doba paměti znamená
 :r1 čas přístupu k jednomu záznamu v paměti
 :r2 doba potřebná pro přenesení 1 KB dat do paměti
 :r3 čas potřebný pro instalaci paměťového modulu
 :r4 doba potřebná pro načtení celé kapacity paměti
:r1 ok
--
Pro paměť s přímým přístupem platí

 :r1 musím se k informaci "pročíst", doba přístupu není konstantní
 :r2 doba přístupu k libovolnému místu v paměti je konstantní
 :r3 obsah z adres nižších hodnot získám rychleji nez vyšších
:r2 ok
--
Energeticky závislá paměť obecně obsahuje po obnově napájení
 :r1 předdefinovaný konstantní obsah
 :r2 samé nuly
 :r3 samé jedničky
 :r4 obsah paměti je nedefinovaný
:r4 ok
--
Energeticky závislá paměť typicky je
 :r1 paměť RAM
 :r2 harddisk
 :r3 paměť Flash
 :r4 CD-R
:r1 ok
--
Správný postup čtení dat z paměti je
 :r1 procesor vloží adresu do adresového registru, příkaz čti, procesor převezme informaci z datového registru
 :r2 procesor vloží adresu do datového registru, příkaz čti, procesor převezme informaci z datového registru
 :r3 procesor vloží adresu do adresového registru, procesor zapíše informaci z datového registru, příkaz čti
 :r4 žádná z uvedených možností neplatí
:r1 ok
--
Paměť určená pro čtení i pro zápis má zkratku
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 RWM
:r4 ok
--
Zpětnému proudu v ROM pamětech zabraňuje
 :r1 použití vodičů
 :r2 použití polovodičové součástky
 :r3 použití nevodičů
 :r4 žádná z uvedených možností
:r2 ok
--
Kolikrát je možno zapisovat do paměti PROM?
 :r1 pouze při výrobě
 :r2 lze jednou naprogramovat
 :r3 lze přeprogramovat libovolněkrát
:r2 ok
--
Ultrafialovým světlem lze přemazat paměť
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 RWM
:r3 ok
--
Elektrickým proudem lze přemazat paměť
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 EEPROM

:r4 ok
--
Paměť, ze které se většinou čte, maže se elektrickým proudem a dá se do ní i zapisovat má zkratku
 :r1 RMM
 :r2 RWM
 :r3 ROM
 :r4 RUM
:r1 ok
--
Pro asociativní paměť <B>ne</B>platí
 :r1 v paměti se plní klíč a obsah
 :r2 paměť klíčů se prohledává paralelně
 :r3 zkratka je CAM
 :r4 používá se jako operační paměť
:r4 ok
--
CAM paměti předám adresu. Nejdříve ji hledá v
 :r1 adresovém registru
 :r2 datovém registru
 :r3 obsahu ke klíčům
 :r4 paměti klíčů
:r4 ok
--
Jaké sběrnice jsou mezi procesorem a pamětí?
 :r1 pouze datová
 :r2 pouze adresová
 :r3 datová a adresová
 :r4 datová, adresová a pro v/v zařízení
:r3 ok
--
Jakou funkci u paměti má refresh cyklus?
 :r1 jednorázově vymaže obsah paměti
 :r2 obnovuje data uložená v dynamické paměti
 :r3 obnovuje data uložená ve statické paměti
 :r4 opraví chybu v paměti
:r2 ok
--
Mezi paměti s výhradně s přímým přístupem patří
 :r1 páska
 :r2 disk
 :r3 operační paměť
:r3 ok
--
Která z uvedených pamětí není programovatelná?
 :r1 ROM
 :r2 PROM
 :r3 EPROM
 :r4 EEPROM
:r1 ok
--
Pro statickou paměť <B>ne</B>platí
 :r1 informace se udržuje, pokud je napájení
 :r2 informace se udržuje, i když není napájení
 :r3 informace se neudržuje, když není napájení
:r2 ok
--
ROM je paměť
 :r1 pouze pro zápis
 :r2 pouze pro čtení
 :r3 pro zápis i pro čtení

 :r4 žádná odpověď není správná
:r2 ok
--
ROM je zkratka pro
 :r1 read only memory
 :r2 read on memory
 :r3 read only matter
 :r4 ride on memory
:r1 ok
--
Páska je paměť
 :r1 se sekvenčním přístupem
 :r2 s přímým přístupem
 :r3 s kombinovaným přístupem
 :r4 s indexsekvenčním přístupem
:r1 ok
--
Na libovolnou adresu v paměti s přímým přístupem se dostanu typicky
 :r1 za proměnlivý čas
 :r2 za konstatntní čas
 :r3 záleží na nastaveni v operačním systému
 :r4 nelze jednoznačně určit
:r2 ok
--
Registr PC -- čítač instrukcí v procesoru obsahuje
 :r1 adresu právě prováděné instrukce.
 :r2 počet již provedených instrukcí.
 :r3 počet instrukcí, které zbývají do konce programu.
:r1 ok
--
Jednou z fází zpracování instrukce procesorem není:
 :r1 výběr operačního kódu z paměti
 :r2 výběr adresy operandu z paměti
 :r3 kopírování instrukce do paměti
 :r4 provedení instrukce
 :r5 zápis výsledků zpracované instrukce
:r3 ok
--
Pro adresaci operační paměti mající kapacitu 64 K adresovatelných jednotek
(bajtů) je třeba adresová sběrnice šířky
 :r1 10 bitů.
 :r2 16 bitů.
 :r3 20 bitů.
 :r4 32 bitů.
:r2 ok
--
Pro adresaci operační paměti mající kapacitu 1 M adresovatelných jednotek
(bajtů) je třeba adresová sběrnice šířky
 :r1 10 bitů.
 :r2 16 bitů.
 :r3 20 bitů.
 :r4 32 bitů.
:r3 ok
--
Pro adresaci uvnitř 4KB stránky potřebuji adresu širokou
 :r1 4 bity.
 :r2 8 bitů.
 :r3 10 bitů.
 :r4 12 bitů.
 :r5 16 bitů.
:r4 ok

--
Pro adresaci 4 MB paměti potřebuji adresu širokou
 :r1 16 bitů
 :r2 18 bitů
 :r3 20 bitů
 :r4 22 bitů
 :r5 24 bitů
:r4 ok
--
Jak široká musí být adresa, pokud chceme adresovat 1 M stránek a každá stránka má velikost 4 K adresovatelných jednotek.
 :r1 12 bitů.
 :r2 16 bitů.
 :r3 22 bitů.
 :r4 32 bitů.
:r4 ok
--
PC -> AR, 0 -> WR, DR -> IR
PC+1 -> AR, 0 -> WR, DR -> TA_L
PC+2 -> AR, 0 -> WR, DR -> TA_H
TA -> AR, 0 -> WR, DR -> A
PC+2 -> PC
 :r1 jsou mikroinstrukce instrukce LDA
 :r2 jsou mikroinstrukce instrukce STA
 :r3 jsou mikroinstrukce jiné instrukce
 :r4 tyto mikroinstrukce jsou nekorektní
:r4 ok
--
Mezi aritmetické instrukce fiktivního procesoru definovaného na přednáškách patří pouze tyto
 :r1 ADD, MOV, CMP
 :r2 STA, ADD, CMA
 :r3 ADD, CMA, INR
:r3 ok
--
Příznaky pro větvení programu vždy nastavují tyto instrukce fiktivního procesoru definovaného na přednáškách
 :r1 ADD, INR
 :r2 LDA, JMP
 :r3 MOV, STA
:r1 ok
--
Pro nastavení příznaků v procesoru definovaném na přednáškách použiji instrukci
 :r1 STA
 :r2 LDA
 :r3 CMP
 :r4 JMP
 :r5 JP
:r3 ok
--
Instrukce mající zkratku LDA typicky znamená
 :r1 ulož obsah registru A do paměti na adresu zadanou operandem instrukce.
 :r2 vynuluj obsah registru A.
 :r3 zvyš obsah registru A o jedničku.
 :r4 naplň obsah registru A hodnotou z paměti.
:r4 ok
--
Instrukce mající zkratku JMP typicky provádí
 :r1 nepodmíněný skok.
 :r2 podmíněný skok na adresu zadanou operandem.

 :r3 volání podprogramu.
:r1 ok
--
Příznakový registr procesoru se používá na
 :r1 sledování výkonnosti procesoru.
 :r2 realizaci podmíněných skoků.
 :r3 zaznamenávání verzí firmware procesoru.
:r2 ok
--
Instrukce CMP pro porovnání typicky
 :r1 větší číslo uloží do registru A.
 :r2 uloží do registru A hodnotu 1, pokud je první číslo větší.
 :r3 pouze nastaví příznaky.
:r3 ok
--
Posloupnost instrukcí
<PRE>
     LDA x
     MOV B,A
     LDA y
     CMP B
     JP ne
ano: ...
     JMP ven
ne: ...
ven: ...
</PRE>
vyjadřuje příkaz
 :r1 IF x>y THEN ano ELSE ne;
 :r2 IF x>=y THEN ano ELSE ne;
 :r3 IF x<y THEN ano ELSE ne;
 :r4 IF x<=y THEN ano ELSE ne;
:r1 ok
--
Posloupnost instrukcí
<PRE>
     LDA y
     MOV B,A
     LDA x
     CMP B
     JM ne
ano: ...
     JMP ven
ne: ...
ven: ...
</PRE>
vyjadřuje příkaz
 :r1 IF x>y THEN ano ELSE ne;
 :r2 IF x>=y THEN ano ELSE ne;
 :r3 IF x<y THEN ano ELSE ne;
 :r4 IF x<=y THEN ano ELSE ne;
:r2 ok
--
Posloupnost instrukcí
<PRE>
     LDA y
     MOV B,A
     LDA x
     CMP B
     JP ne
ano: ...

```
     JMP ven
ne:  ...
ven: ...
</PRE>
vyjadřuje příkaz
 :r1 IF x&gt;y THEN ano ELSE ne;
 :r2 IF x&gt;=y THEN ano ELSE ne;
 :r3 IF x&lt;y THEN ano ELSE ne;
 :r4 IF x&lt;=y THEN ano ELSE ne;
:r3 ok
--
Posloupnost instrukcí
<PRE>
     LDA x
     MOV B,A
     LDA y
     CMP B
     JM ne
ano: ...
     JMP ven
ne:  ...
ven: ...
</PRE>
vyjadřuje příkaz
 :r1 IF x&gt;y THEN ano ELSE ne;
 :r2 IF x&gt;=y THEN ano ELSE ne;
 :r3 IF x&lt;y THEN ano ELSE ne;
 :r4 IF x&lt;=y THEN ano ELSE ne;
:r4 ok
--
PC -&gt; AR, 0 -&gt; WR, DR -&gt; IR
PC+1 -&gt; AR, 0 -&gt; WR, DR -&gt; TA<sub>L</sub>
PC+2 -&gt; AR, 0 -&gt; WR, DR -&gt; TA<sub>H</sub>
TA -&gt; AR, 0 -&gt; WR, DR -&gt; TAX<sub>L</sub>
TA+1 -&gt; AR, 0 -&gt; WR, DR -&gt; TAX<sub>H</sub>
TAX -&gt; AR, A -&gt; DR, 1 -&gt; WR
PC+3 -&gt; PC
 :r1 jsou mikroinstrukce instrukce LDA
 :r2 jsou mikroinstrukce instrukce STA
 :r3 jsou mikroinstrukce LDAX (nepřímé naplnění)
 :r4 jsou mikroinstrukce STAX (nepřímé naplnění)
 :r5 tyto mikroinstrukce jsou nekorektní
:r4 ok
--
Instrukce podmíněného skoku
 :r1 provede následující instrukci, pokud je splněna podmínka.
 :r2 skočí na instrukci, jejíž adresa je zadána operandem, pokud podmínka
není splněna.
 :r3 provede následující instrukci, pokud podmínka splněna není.
:r3 ok
--
Operace PUSH nad zásobníkem
 :r1 vloží položku do zásobníku.
 :r2 vybere položku ze zásobníku.
 :r3 stlačí obsah zásobníku.
:r1 ok
--
Instrukce PUSH osmibitového procesoru definovaného na přednáškách vkládá do
zásobníku
 :r1 1 bajt
 :r2 1 dvoubajtové slovo
```

 :r3 2 dvoubajtová slova
 :r4 1 bit
:r2 ok
--
Jaký je správný postup operací?
 :r1 PUSH sníží SP a uloží položku na adresu podle SP; POP vybere z adresy podle SP a zvýší SP.
 :r2 PUSH sníží SP a uloží položku na adresu podle SP; POP zvýší SP a vybere z adresy podle SP.
 :r3 PUSH uloží položku na adresu podle SP a sníží SP; POP vybere z adresy podle SP a zvýší SP.
:r1 ok
--
Instrukce volání podprogramu musí
 :r1 uchovat návratovou adresu.
 :r2 uchovat obsah čítače instrukcí.
 :r3 uchovat obsah registrů do zásobníku.
:r1 ok
--
Pojem 'time-out' při provádění V/V operací znamená, že např.
 :r1 zahájená výstupní operace neodpověděla 'hotovo' do definované doby.
 :r2 mezi výstupní a vstupní operací musí být prodleva definované doby.
 :r3 před zahájením vstupní operace lze signál 'start' poslat ne dříve než uplyne definovaná doba.
:r1 ok
--
Posloupnost instrukcí
<PRE>
       START
opak: FLAG opak
       IN
       STA x
</PRE>
je podle toho, jak jsme si na přednáškách definovali vlastní procesor (pomíjíme otázku time-outu, neefektního využití procesoru),
 :r1 korektní operace čtení ze vstupního zařízení
 :r2 korektní operace zápisu do výstupního zařízení
 :r3 žádná z ostatních odpovědí není správná
:r1 ok
--
Posloupnost instrukcí
<PRE>
       LDA x
       START
       OUT
opak: FLAG opak
</PRE>
je podle toho, jak jsme si na přednáškách definovali vlastní procesor (pomíjíme otázku time-outu, neefektního využití procesoru),
 :r1 korektní operace čtení ze vstupního zařízení
 :r2 korektní operace zápisu do výstupního zařízení
 :r3 žádná z ostatních odpovědí není správná
:r3 ok
--
Ve kterém z následujících okamžiků by mělo dojít ke vzniku přerušení?
 :r1 zahájení tisku znaku
 :r2 konec tisku znaku
 :r3 ukončení programu
:r2 ok
--

Které z konstatování vztahujících se k okamžiku přerušení procesu je <B>ne</B>správné?
 :r1 Přerušit nelze během provádění instrukce.
 :r2 Přerušit lze pouze tehdy, je-li to povoleno (nejde-li o nemaskovatelné přerušení).
 :r3 Přerušit nelze bezprostředně po zahájení obsluhy přerušení.
 :r4 Přerušení nastane ihned po žádosti signálem INTERRUPT.
:r4 ok
--
Jaké je správné modelové chování obsluhy vzniku přerušení?
 :r1 Mikroinstrukce musí uložit PC a vynulovat IF. Programem se ukládají všeobecné registry.
 :r2 Mikroinstrukce musí uložit PC a všeobecné registry. Program dle svého zvážení vynuluje IF.
 :r3 Mikroinstrukce uloží obsah PC. Program uloží dle zvážení obsah všeobecných registrů a vynuluje IF.
:r1 ok
--
Operační kód (operační znak) je
 :r1 numerické vyjádření konkrétní instrukce, je vždy stejně dlouhý
 :r2 numerické vyjádření konkrétní instrukce, má typicky proměnlivou délku
 :r3 je adresa operandu
 :r4 je adresa 1. a 2. operandu
:r2 ok
--
Operační kód není
 :r1 operační znak
 :r2 numerické vyjádření konkrétní instrukce, které má proměnlivou délku
 :r3 součást instrukce
 :r4 žádná z uvedených možností
:r4 ok
--
Pro čítač instrukcí procesoru <B>ne</B>platí
 :r1 může mít zkratku PC
 :r2 může mít zkratku IP
 :r3 obsahuje adresu prováděné instrukce
 :r4 žádná z uvedených možností
:r4 ok
--
Která instrukce naplní registr A obsahem slabiky z paměti?
 :r1 STA
 :r2 LDA
 :r3 INA
 :r4 JMP
:r2 ok
--
Instrukce STA
 :r1 uloží registr A do paměti
 :r2 naplní registr A obsahem slabiky z paměti
 :r3 je nepodmíněný skok na adresu A
 :r4 žádná z uvedených možností
:r1 ok
--
Instrukce JMP je
 :r1 nepodmíněný skok
 :r2 podmíněný skok
 :r3 uloží registr P do paměti
 :r4 žádná z uvedených možností
:r1 ok
--
Osmibitový procesor se 64KB pamětí má

 :r1 8bitovou datovou sběrnici a 20bitovou adresovou sběrnici
 :r2 8bitovou datovou sběrnici a 8bitovou adresovou sběrnici
 :r3 8bitovou datovou sběrnici a 16bitovou adresovou sběrnici
:r3 ok
--
Registr PC procesoru naplníme instrukcí
 :r1 LDA
 :r2 STA
 :r3 JMP
 :r4 žádnou z uvedených
:r3 ok
--
Pomocný 16bitový registr TA procesoru definovaného na přednáškách se skládá
z
 :r1 8bitového TA High a 8bitového TA Low
 :r2 12bitového TA High a 4bitového TA Low
 :r3 4bitového TA High a 12bitového TA Low
 :r4 žádná z uvedených možností
:r1 ok
--
První fází každé instrukce je
 :r1 výběr operandu
 :r2 provedení instrukce
 :r3 výběr operačního znaku
 :r4 aktualizace PC
:r3 ok
--
Pro mikroinstrukci výběr operačního znaku <B>ne</B>platí
 :r1 cílem je vložit do instrukčního registru instrukci
 :r2 je vždy 1. fází instrukce
 :r3 cílem je vložit do datového registru data
 :r4 je součástí např. instrukce LDA
:r3 ok
--
Mikroinstrukce výběr operačního znaku znamená
 :r1 procesor zjistí, kterou instrukci provádí
 :r2 procesor načte adresu z adresového registru
 :r3 procesor zahájí instrukci LDA
 :r4 žádná z uvedených možností
:r1 ok
--
Mezi mikroinstrukce instrukce LDA <B>ne</B>patří
 :r1 výběr operačního znaku
 :r2 výběr operandu
 :r3 aktualizace registru PC zvýšením o délku instrukce
 :r4 naplnění registru PC hodnotou operandu instrukce
:r4 ok
--
Instrukce INR procesoru definovaného na přednáškách způsobí
 :r1 zvýší obsah registru o jedna
 :r2 sníží obsah registru o jedna
 :r3 uloží obsah registru R do paměti
 :r4 načte obsah registru R z paměti
:r1 ok
--
Instrukce CMA procesoru definovaného na přednáškách způsobí
 :r1 inverzi bitů v registru A
 :r2 zvýší obsah registru A o jedna
 :r3 sníží obsah jedničku A o jedna
 :r4 žádná z uvedených možností
:r1 ok

--
Která instrukce sníží obsah registru o jedna
 :r1 INR
 :r2 CMA
 :r3 ADD
 :r4 žádná z uvedených možností
:r4 ok
--
Instrukce ADD procesoru definovaného na přednáškách
 :r1 přičte obsah registru k registru A
 :r2 invertuje bity v registru A
 :r3 vždy zvýší obsah registru A o jedna
 :r4 žádná z uvedených možností
:r1 ok
--
Příznak procesoru definovaného na přednáškách není
 :r1 jednobitový indikátor
 :r2 Z (zero)
 :r3 CY (Carry)
 :r4 žádná z uvedených možností
:r4 ok
--
S (Sign) je příznak procesoru definovaného na přednáškách, kterým je
 :r1 kopie znaménkového bitu výsledku operace
 :r2 kopie znaménkového bitu 1. operandu
 :r3 kopie znaménkového bitu 2. operandu
 :r4 1 při nulovém výsledku operace
:r1 ok
--
Pro příznaky procesoru definovaného na přednáškách platí
 :r1 nastavuje je programátor
 :r2 nastavuje je procesor
 :r3 nastavuje je procesor a programátor může nastavování vypnout
 :r4 žádná z uvedených možností
:r2 ok
--
Instrukce procesoru definovaného na přednáškách CMP B porovná obsah
registru A s obsahem registru B a
 :r1 změní podle toho příznaky
 :r2 nezmění podle toho příznaky
 :r3 uloží výsledek do registru A
 :r4 uloží výsledek do registru B
:r1 ok
--
Mezi příznaky procesoru definovaného na přednáškách <B>ne</B>patří
 :r1 CY
 :r2 AC
 :r3 TA
 :r4 Z
:r3 ok
--
Změnu znaménka u čísla v registru A procesoru definovaného na přednáškách
provedeme posloupností instrukcí
 :r1 CMA, INR A
 :r2 CMA, MOV B,A
 :r3 INR A, CMA
 :r4 žádná z uvedených možností
:r1 ok
--
Pro zásobník procesoru definovaného na přednáškách <B>ne</B>platí, že
 :r1 je datová struktura fungující systémem LIFO

 :r2 je datová struktura fungující systémem FIFO
 :r3 vkládá se do ní operací PUSH
 :r4 vybírá se z ní operací POP
:r2 ok
--
PUSH procesoru definovaného na přednáškách
 :r1 je instrukce, vkládá obsah registru do zásobníku
 :r2 je instrukce, vybírá obsah ze zásobníku
 :r3 je příznak
 :r4 je interní registr
:r1 ok
--
PSW procesoru definovaného na přednáškách je
 :r1 stavové slovo procesoru, tvořeno z registru A a příznaků
 :r2 stavové slovo procesoru, tvořeno z registru A
 :r3 stavové slovo procesoru, tvořeno z příznaku na předdefinovaný registr
 :r4 žádná z uvedených možností
:r1 ok
--
Pro zásobník procesoru definovaného na přednáškách platí
 :r1 má kontrolu podtečení
 :r2 nemá kontrolu podtečení
 :r3 je strukturou First in First out
 :r4 žádná z uvedených možností
:r2 ok
--
LXISP procesoru definovaného na přednáškách
 :r1 je ukazatel na vrchol zásobníku
 :r2 zapíše hodnotu na dno zásobníku
 :r3 definuje dno zásobníku
 :r4 instrukce, která vkládá obsah registru A do zásobníku
:r3 ok
--
Instrukce PUSH procesoru definovaného na přednáškách
 :r1 numericky snižuje ukazatel vrcholu zásobníku
 :r2 numericky zvyšuje ukazatel vrcholu zásobníku
 :r3 inkrementuje SP
 :r4 žádná z uvedených možností
:r1 ok
--
Instrukce POP procesoru definovaného na přednáškách
 :r1 definuje dno zásobníku
 :r2 snižuje ukazatel vrcholu zásobníku
 :r3 dekrementuje SP
 :r4 žádná z uvedených možností
:r4 ok
--
Pro instrukci RET procesoru definovaného na přednáškách <B>ne</B>platí
 :r1 vrátí se z podprogramu do těla programu
 :r2 obsah vrcholu zásobníku je vložen do registru PC
 :r3 vrátí se na absolutní začátek programu
 :r4 používá se na konci podprogramu
:r3 ok
--
Která posloupnost instrukcí může korektně obsloužit time-out při
programování V/V operace procesoru definovaného na přednáškách
 :r1 100 START
101 INR
102 JZ 108
105 FLAG 101
 :r2 100 START

101 INR
102 JZ 101
105 FLAG 101
 :r3 100 START
101 INR
102 FLAG 101
105 JZ 102
:r1 ok
--
Instrukce OUT procesoru definovaného na přednáškách
 :r1 zapíše obsah reg. A na datovou sběrnici pro v/v zařízení
 :r2 načte obsah datové sběrnice od v/v zařízení a uloží jej do A
 :r3 zapíše obsah reg. A a zahájí vstupně výstuní operaci
:r1 ok
--
Která instrukce procesoru definovaného na přednáškách skočí na adresu,
není-li operace hotova?
 :r1 START
 :r2 FLAG
 :r3 IN
 :r4 OUT
:r2 ok
--
Posloupnost instrukcí procesoru definovaného na přednáškách
LDA x, OUT, START, FLAG
je
 :r1 korektní operace čtení ze vstupního zařízení
 :r2 korektní operace zápisu do výstupního zařízení
 :r3 žádná z ostatních odpovědí není správná
:r2 ok
--
Posloupnost instrukcí procesoru definovaného na přednáškách
START, IN, STA x, FLAG
je
 :r1 korektní operace čtení ze vstupního zařízení
 :r2 korektní operace zápisu do výstupního zařízení
 :r3 žádná z ostatních odpovědí není správná
:r3 ok
--
Co je time-out?
 :r1 doba, kterou jsme ochotni čekat na dokončení V/V operace
 :r2 doba, kterou jsme ochotni čekat na začátek V/V operace
 :r3 doba, kterou nemůžeme ovlivnit (je předdefinovaná)
:r1 ok
--
Signál INTERRUPT (INTR)
 :r1 žádá o přerušení v procesoru
 :r2 deaktivuje rutinu pro obsluhu přerušení
 :r3 žádá o ukončení provádění procesu
 :r4 žádá o uvedení procesoru do počátečních podmínek
:r1 ok
--
Která činnost se vykonává jako poslední při návratu z přerušení procesoru
definovaného na přednáškách?
 :r1 provedení obslužné rutiny, která zjistí kdo žádá o přerušení
 :r2 přerušení provádění programu
 :r3 obnovení PC, A, ...
 :r4 úklid obsahu registrů PC, A, ...
:r3 ok
--
Pro přerušení platí:

 :r1 přerušit lze pouze během provádění instrukce
 :r2 lze přerušit bezprostředně po zahájení obsluhy předchozího přerušení
 :r3 o přerušení se musí požádat signálem INTERRUPT
 :r4 přerušení se používá typicky v kritické sekci
:r3 ok
--
Instrukce, která zakáže přerušení procesoru definovaného na přednáškách, se nazývá
 :r1 STI
 :r2 CLI
 :r3 INTERRUPT
 :r4 žádná možnost není správná
:r2 ok
--
Co je v registru PC procesoru definovaného na přednáškách při uplatnění žádosti o přerušení
 :r1 adresa instrukce, která byla provedena před přerušením
 :r2 adresa instrukce, která nebyla provedena v důsledku přerušení
 :r3 adresa vrcholu zásobníku
:r2 ok
--
Během uplatnění přerušení <B>není</B> provedeno
 :r1 uložení registru PC do zásobníku
 :r2 vynulování IF
 :r3 povolení přerušení
 :r4 uklizení registru A a dalších do zásobníku
:r3 ok
--
Která z instrukcí <B>ne</B>patří mezi instrukce procesoru definovaného na přednáškách, které se použijí při návratu z přerušení
 :r1 POP
 :r2 STI
 :r3 RET
 :r4 CLI
:r4 ok
--
Co <B>ne</B>platí pro instrukci STI procesoru definovaného na přednáškách
 :r1 povolí přerušení až po provedení následující instrukce
 :r2 nastaví IF na hodnotu 1
 :r3 povolí přerušení po svém dokončení
:r3 ok
--
Signál RESET procesoru definovaného na přednáškách <B>ne</B>způsobí
 :r1 nastavení procesoru do počátečních podmínek
 :r2 vynulování příznaků procesoru
 :r3 předání řízení na adresu ukazující zpravidla do permanentní paměti
 :r4 zakázání přerušení
 :r5 vynulování IF
:r2 ok
--
Pro signál RESET procesoru definovaného na přednáškách <B>ne</B>platí
 :r1 provede se kdykoliv
 :r2 nastaví IF na nulu
 :r3 provede se pouze při přerušení
 :r4 předá řízení na adresu ukazující zpravidla do v permanentní paměti
:r3 ok
--
Výběr instrukcí procesoru definovaného na přednáškách je řízen registrem
 :r1 PC
 :r2 AR
 :r3 DR

 :r4 IR
:r1 ok
--
Který z registrů procesoru definovaného na přednáškách není 16bitový
 :r1 PC
 :r2 IR
 :r3 TA
 :r4 AR
:r2 ok
--
Která instrukce procesoru definovaného na přednáškách nastavuje příznaky
 :r1 LDA
 :r2 ADD
 :r3 STA
 :r4 JMP
:r2 ok
--
Která instrukce procesoru definovaného na přednáškách porovná zadaný
registr s registrem A
 :r1 CMA
 :r2 CMP
 :r3 STA
 :r4 LDA
:r2 ok
--
Zásobník má strukturu
 :r1 LIFO
 :r2 FIFO
 :r3 PIFO
 :r4 SIFO
:r1 ok
--
Fronta má strukturu
 :r1 LIFO
 :r2 FIFO
 :r3 PIFO
 :r4 SIFO
:r2 ok
--
Pro instrukci CALL procesoru definovaného na přednáškách <B>ne</B>platí
 :r1 uloží návratovou adresu do zásobníku
 :r2 provede nepodmíněný skok na zadanou adresu
 :r3 přečte obsah zadaného registru
 :r4 provede totéž co posloupnost instrukcí PUSH a JMP
:r3 ok
--
Procesor rozlišuje komunikaci s pamětí a se V/V zařízeními
 :r1 užíváním různých sběrnic
 :r2 signálem M/IO
 :r3 signálem NMI
 :r4 signálem CLK
:r2 ok
--
Jak široká musí být adresa, pokud chceme adresovat 1 K stránek a každá
stránka má velikost 4 K adresovatelných jednotek.
 :r1 12 bitů.
 :r2 16 bitů.
 :r3 22 bitů.
 :r4 32 bitů.
:r3 ok
--

Pokud používáme virtualizaci paměti, pak
 :r1 šířka virtuální adresy by měla být větší nebo rovna šířce reálné adresy.
 :r2 šířka virtuální adresy by měla být menší nebo rovna šířce reálné adresy.
 :r3 se musí šířka virtuální adresy a reálné adresy shodovat.
:r1 ok
--
K obecnému mechanismu virtuální paměti: Co je obvyklé?
 :r1 Počet stránek je větší než počet rámců.
 :r2 Počet stránek je roven počtu rámců.
 :r3 Počet stránek je menší než počet rámců.
:r1 ok
--
K obecnému mechanismu virtuální paměti: Která z adres může být širší (má se na mysli, že je více bitová)
 :r1 reálná
 :r2 virtuální
 :r3 bezpodmínečně musí být reálná a virtuální adresa stejně velké
:r2 ok
--
K obecnému mechanismu virtuální paměti: Co platí?
 :r1 Rámce jsou uloženy na disku, stránky jsou v reálné paměti.
 :r2 Stránky jsou uloženy na disku, rámce jsou v reálné paměti.
:r2 ok
--
K obecnému mechanismu algoritmu LRU: Algoritmus LRU vybírá
 :r1 nejdéle nepoužitou položku
 :r2 nejméněkrát použitou položku
 :r3 nejdéle uloženou položku
:r1 ok
--
Algoritmus LRU pro výběr oběti např. při virtualizaci paměti vybírá
 :r1 nejméněkrát použitý obsah rámce.
 :r2 nejdéle nepoužitý obsah rámce.
 :r3 náhodný rámec.
 :r4 předchozí použitý rámec.
:r2 ok
--
Při virtualizaci paměti se používají pojmy
 :r1 segment a stránka.
 :r2 rámec a stránka.
 :r3 segment a rámec.
:r2 ok
--
Špinavá stránka
 :r1 je stránka odložená do virtuální paměti
 :r2 je stránka změněná v reálné paměti
 :r3 je stránka s porušenou strukturou
 :r4 je stránka s neprivilegovaným instrukčním kódem
:r2 ok
--
K obecnému mechanismu algoritmu LRU: K úplnému ošetření osmi položek algoritmem LRU (pomocí neúplné matice) bychom potřebovali kolik bitů v neúplné matici?
 :r1 28
 :r2 36
 :r3 24
 :r4 16
 :r5 8
:r1 ok

--
K obecnému mechanismu algoritmu LRU: K úplnému ošetření šesti položek algoritmem LRU (pomocí neúplné matice) bychom potřebovali kolik bitů v neúplné matici?
 :r1 28
 :r2 21
 :r3 15
 :r4 16
 :r5 8
:r3 ok
--
Pro virtualizaci paměti <B>ne</B>platí
 :r1 paměť dělíme do rámců a disk na stránky
 :r2 reálná adresa ukazuje do reálné paměti
 :r3 počet stránek je větší nebo roven počtu rámců
 :r4 rámec není stejně velký prostor jako stránka
:r4 ok
--
Při virtualizaci paměti <B>ne</B>platí
 :r1 obsah špinavého rámce musím před jeho smazáním zapsat na disk
 :r2 označení čistý rámec odpovídá označení rámec, do kterého nebylo zapsáno
 :r3 rámec je špinavý, pokud má příznak parity nastaven na jedničku
 :r4 do špinavého rámce bylo něco zapsáno
:r3 ok
--
Pro virtualizaci paměti se používá
 :r1 segment
 :r2 stránka
:r2 ok
--
Co platí pro segmenty a stránky:
 :r1 segmenty jsou různé velikosti, stránky jsou stejné velikosti
 :r2 segmenty jsou stejné velikosti, stránky jsou různé velikosti
 :r3 segmenty jsou různé velikosti, stránky jsou různé velikosti
 :r4 segmenty jsou stejné velikosti, stránky jsou stejné velikosti
:r1 ok
--
Co znamená LRU:
 :r1 least recently used
 :r2 last record used
 :r3 load record unsaved
 :r4 let ring upset
:r1 ok
--
Jaká je <B>ne</B>správná konfigurace virtuální paměti u obecného procesoru?
 :r1 32bitová reálná adresa a 48bitová virtuální adresa
 :r2 32bitová reálná adresa a 24bitová virtuální adresa
 :r3 24bitová reálná adresa a 24bitová virtuální adresa
:r2 ok

Převeďte celé kladné číslo 11110000 z dvojkové soustavy do desítkové soustavy. Jaký je správný výsledek?
 :r1 109
 :r2 135
 :r3 141
 :r4 142
 :r5 177
 :r6 240
:r6 ok
--
Převeďte celé kladné číslo 00101010 z dvojkové soustavy do desítkové soustavy. Jaký je správný výsledek?
 :r1 20
 :r2 42
 :r3 104
 :r4 133
 :r5 153
 :r6 165
:r2 ok
--
Převeďte celé kladné číslo 153 z desítkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 01010101
 :r2 10000011
 :r3 10011001
 :r4 10110001
 :r5 11000011
 :r6 11000110
:r3 ok
--
Převeďte celé kladné číslo 202 z desítkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 00100011
 :r2 00110011
 :r3 01110100
 :r4 10011001
 :r5 11001010
 :r6 11010000
:r5 ok
--
Převeďte celé kladné číslo 11111001010110000 z dvojkové soustavy do osmičkové soustavy. Jaký je správný výsledek?
 :r1 032576
 :r2 371260
 :r3 460753
 :r4 546731
 :r5 637512
 :r6 712035
:r2 ok
--
Převeďte celé kladné číslo 111100101010011 z dvojkové soustavy do osmičkové soustavy. Jaký je správný výsledek?
 :r1 074523
 :r2 172530
 :r3 205316
 :r4 230516
 :r5 541207
 :r6 620417
:r1 ok
--

Převeďte celé kladné číslo 073562 z osmičkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 10101111001011
 :r2 111011101110010
 :r3 101000111100001110
 :r4 101010111000001011
 :r5 111000010110001101
 :r6 111010001011101000
:r2 ok
--
Převeďte celé kladné číslo 156732 z osmičkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 1101110111011010
 :r2 10100011001101111
 :r3 101000111100001110
 :r4 110011000100010001
 :r5 110101111000011010
 :r6 1111101011000010 10
:r1 ok
--
Převeďte celé kladné číslo 1101010010101000010 z dvojkové soustavy do šestnáctkové soustavy. Jaký je správný výsledek?
 :r1 ce47b9
 :r2 fd16ab
 :r3 f2bd75
 :r4 06a542
 :r5 69b1d0
 :r6 126083
:r4 ok
--
Převeďte celé kladné číslo 10100011000101101101 00 z dvojkové soustavy do šestnáctkové soustavy. Jaký je správný výsledek?
 :r1 efa372
 :r2 ec4f06
 :r3 d52f7b
 :r4 28c5b4
 :r5 781cfd
 :r6 16458a
:r4 ok
--
Převeďte celé kladné číslo cf6182 z šestnáctkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 1001011001101100 11011
 :r2 1010110010100100 00010
 :r3 1010111000010011111111100
 :r4 1100111101100001100000 10
 :r5 1100111111110101011100 10
 :r6 1111111011001010000010 00
:r4 ok
--
Převeďte celé kladné číslo 7f6b4a z šestnáctkové soustavy do dvojkové soustavy. Jaký je správný výsledek?
 :r1 1110111111000000010100
 :r2 1111100111110110000001 1
 :r3 1111111011010110100 1010
 :r4 1100010110101110100 10010
 :r5 1100011011011000000001111
 :r6 1100011011100011001011 11
:r3 ok
--

Sečtěte dvojková čísla
<TT>01101011</TT>
<TT>11011101</TT>
Jaký je správný výsledek ve dvojkové soustavě?
 :r1 101000
 :r2 10001110
 :r3 10101111
 :r4 101001000
 :r5 110111010
 :r6 111101010
:r4 ok
--
Sečtěte dvojková čísla
<TT>00110000</TT>
<TT>01010100</TT>
Jaký je správný výsledek ve dvojkové soustavě?
 :r1 1101010
 :r2 10000100
 :r3 10000101
 :r4 10100110
 :r5 111001000
 :r6 111001111
:r2 ok
--
Jaký je správný výsledek výrazu 2⁻²?
 :r1 0,125
 :r2 0,25
 :r3 0,0625
 :r4 0,03125
 :r5 0,5
:r2 ok
--
Jaký je správný výsledek výrazu 2⁻¹?
 :r1 0,125
 :r2 0,25
 :r3 0,0625
 :r4 0,03125
 :r5 0,5
:r5 ok
--
Převeďte desetinné číslo 0,74 z desítkové soustavy do dvojkové soustavy na maximálně 6 dvojkových řádů. Jaký je správný výsledek?
 :r1 0,101111
 :r2 0,011
 :r3 0,0101
 :r4 0,001001
 :r5 0,111011
 :r6 0,01
:r1 ok
--
Převeďte desetinné číslo 0,94 z desítkové soustavy do dvojkové soustavy na maximálně 6 dvojkových řádů. Jaký je správný výsledek?
 :r1 0,11111
 :r2 0,01111
 :r3 0,010101
 :r4 0,1111
 :r5 0,1011
 :r6 0,1
:r4 ok
--
Převeďte desetinné číslo 0,111 z dvojkové soustavy do desítkové soustavy na maximálně 4 desítkové řády. Jaký je správný výsledek?
 :r1 0,8125
 :r2 0,0625
 :r3 0,3125

 :r4 0,625
 :r5 0,75
 :r6 0,875
:r6 ok
--
Převeďte desetinné číslo 0,011 z dvojkové soustavy do desítkové soustavy na maximálně 4 desítkové řády. Jaký je správný výsledek?
 :r1 0,1875
 :r2 0,8125
 :r3 0,25
 :r4 0,125
 :r5 0,375
 :r6 0,0625
:r5 ok
--
Převeďte celé číslo -57 z desítkové soustavy do dvojkové soustavy v přímém kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10101101
 :r2 10111001
 :r3 10111110
 :r4 11000001
 :r5 11010100
 :r6 11100011
:r2 ok
--
Převeďte celé číslo -49 z desítkové soustavy do dvojkové soustavy v přímém kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10001111
 :r2 10011001
 :r3 10110001
 :r4 10111001
 :r5 11010000
 :r6 11110100
:r3 ok
--
Převeďte celé číslo 10101101 ze dvojkové soustavy v přímém kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -122
 :r2 -61
 :r3 -55
 :r4 -45
 :r5 -16
 :r6 -2
:r4 ok
--
Převeďte celé číslo 11010100 ze dvojkové soustavy v přímém kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -122
 :r2 -104
 :r3 -84
 :r4 -81
 :r5 -29
 :r6 -2
:r3 ok
--
Převeďte celé číslo -50 z desítkové soustavy do dvojkové soustavy v inverzním kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10000010
 :r2 10001111
 :r3 10111000
 :r4 11000010

 :r5 11001100
 :r6 11001101
:r6 ok
--
Převeďte celé číslo -4 z desítkové soustavy do dvojkové soustavy v inverzním kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10000100
 :r2 10010001
 :r3 11001100
 :r4 11001110
 :r5 11100001
 :r6 11111011
:r6 ok
--
Převeďte celé číslo 10011010 ze dvojkové soustavy v inverzním kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -127
 :r2 -107
 :r3 -101
 :r4 -53
 :r5 -38
 :r6 -5
:r3 ok
--
Převeďte celé číslo 10001000 ze dvojkové soustavy v inverzním kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -119
 :r2 -105
 :r3 -44
 :r4 -21
 :r5 -14
 :r6 -3
:r1 ok
--
Převeďte celé číslo -60 z desítkové soustavy do dvojkové soustavy ve dvojkovém doplňkovém kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10011011
 :r2 10011100
 :r3 10100000
 :r4 10100101
 :r5 10110100
 :r6 11000100
:r6 ok
--
Převeďte celé číslo -110 z desítkové soustavy do dvojkové soustavy ve dvojkovém doplňkovém kódu na 8 bitech. Jaký je správný výsledek?
 :r1 10010010
 :r2 10011000
 :r3 10101101
 :r4 10110011
 :r5 11100000
 :r6 11111100
:r1 ok
--
Převeďte celé číslo 11001000 ze dvojkové soustavy ve dvojkovém doplňkovém kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -111
 :r2 -99
 :r3 -56
 :r4 -23
 :r5 -8

 :r6 -7
:r3 ok
--
Převeďte celé číslo 10111101 ze dvojkové soustavy ve dvojkovém doplňkovém kódu na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -99
 :r2 -90
 :r3 -72
 :r4 -67
 :r5 -58
 :r6 -32
:r4 ok
--
Sečtěte 8bitově dvojková čísla ve dvojkovém doplňkovém kódu:
<TT>00101000</TT>
<TT>11001001</TT>
Obdržíte výsledek:
<TT>11110001</TT>
Došlo k přeplnění?
 :r1 Ne.
 :r2 Ano.
:r1 ok
--
Sečtěte 8bitově dvojková čísla ve dvojkovém doplňkovém kódu:
<TT>11101100</TT>
<TT>01110001</TT>
Obdržíte výsledek:
<TT>01011101</TT>
Došlo k přeplnění?
 :r1 Ne.
 :r2 Ano.
:r1 ok
--
Sečtěte 8bitově dvojková čísla v inverzním kódu:
<TT>11111110</TT>
<TT>01011011</TT>
Obdržíte výsledek:
<TT>01011010</TT>
Byl aplikován kruhový přenos?
 :r1 Ne.
 :r2 Ano.
:r2 ok
--
Sečtěte 8bitově dvojková čísla v inverzním kódu:
<TT>01001001</TT>
<TT>10100001</TT>
Obdržíte výsledek:
<TT>11101010</TT>
Byl aplikován kruhový přenos?
 :r1 Ne.
 :r2 Ano.
:r1 ok
--
Máte dvě osmibitová dvojková čísla ve dvojkovém doplňkovém kódu. Od prvního čísla odečtěte 8bitově druhé číslo:
<TT>01010010</TT>
<TT>00101111</TT>
Jaký je správný výsledek ve dvojkové soustavě?
 :r1 00000010
 :r2 00000011
 :r3 00010001
 :r4 00100011
 :r5 10111100
 :r6 11000001
:r4 ok
--
Máte dvě osmibitová dvojková čísla ve dvojkovém doplňkovém kódu. Od prvního čísla odečtěte 8bitově druhé číslo:
<TT>11001011</TT>
<TT>11011110</TT>
Jaký je správný výsledek ve dvojkové soustavě?
 :r1 00011111
 :r2 01110100
 :r3 10010000
 :r4 10011101

```
 :r5 11010100
 :r6 11101101
:r6 ok
--
Jak je v UTF-8 binárně zobrazen znak 'Ň' U+0147?
 :r1 10010100 11000111
 :r2 11000101 10000111
 :r3 11010010 11000111
 :r4 11010111 11000111
 :r5 11011001 11000111
 :r6 11111000 10000111
:r2 ok
--
Jak je v UTF-8 binárně zobrazen znak 'č' U+010d?
 :r1 10000101 11001101
 :r2 10011100 10001101
 :r3 10100110 11001101
 :r4 11000100 10001101
 :r5 11011110 11001101
 :r6 11100001 11001101
:r4 ok
--
Jakému znaku v UTF-8 odpovídá binární hodnota 11000101 10000111 ?
 :r1 'ý' U+00fd
 :r2 'Ť' U+0164
 :r3 'ů' U+016f
 :r4 'Ě' U+011a
 :r5 'Ď' U+010e
 :r6 'Ň' U+0147
:r6 ok
--
Jakému znaku v UTF-8 odpovídá binární hodnota 11000011 10001101 ?
 :r1 'Ř' U+0158
 :r2 'ý' U+00fd
 :r3 'Í' U+00cd
 :r4 'ó' U+00f3
 :r5 'š' U+0161
 :r6 'Ě' U+011a
:r3 ok
--
Převeďte celé číslo 49 z desítkové soustavy do dvojkové soustavy v kódu
posunuté nuly na 8 bitech. Jaký je správný výsledek?
 :r1 01101000
 :r2 01111011
 :r3 10010111
 :r4 10110000
 :r5 11110011
 :r6 11111100
:r4 ok
--
Převeďte celé číslo -9 z desítkové soustavy do dvojkové soustavy v kódu
posunuté nuly na 8 bitech. Jaký je správný výsledek?
 :r1 00100001
 :r2 01010010
 :r3 01010101
 :r4 01100110
 :r5 01110110
 :r6 10110111
:r5 ok
--
```

Převeďte celé číslo 00011011 ze dvojkové soustavy v kódu posunuté nuly na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -110
 :r2 -100
 :r3 -41
 :r4 -12
 :r5 110
 :r6 120
:r2 ok
--
Převeďte celé číslo 11100100 ze dvojkové soustavy v kódu posunuté nuly na 8 bitech do desítkové soustavy. Jaký je správný výsledek?
 :r1 -105
 :r2 -98
 :r3 65
 :r4 79
 :r5 101
 :r6 123
:r5 ok
--
Převeďte desetinné číslo 1520,0 z desítkové soustavy do dvojkové soustavy v kódu IEEE 754 s 8bitovým exponentem. Jaký je správný výsledek?
 :r1 00010010 10001110 01000100 00110110
 :r2 00110100 10011100 00110001 00110010
 :r3 01000100 00100110 01100111 00010000
 :r4 01000100 10111110 00000000 00000000
 :r5 01011011 11101011 00010011 01111010
 :r6 01101100 10111111 11000011 10001100
:r4 ok
--
Převeďte desetinné číslo -38,6875 z desítkové soustavy do dvojkové soustavy v kódu IEEE 754 s 8bitovým exponentem. Jaký je správný výsledek?
 :r1 10000110 01011010 01011100 00111100
 :r2 10101101 11110110 00010101 01011101
 :r3 11000010 00011010 11000000 00000000
 :r4 11000111 11011010 11011100 11111011
 :r5 11101101 00001101 01100101 00010110
 :r6 11111111 01101100 10100001 00011010
:r3 ok
--
Převeďte desetinné číslo v kódu IEEE 754 s 8bitovým exponentem
01000001 01111001 00000000 00000000
ze dvojkové soustavy do desítkové soustavy. Jaký je správný výsledek?
 :r1 13,1875
 :r2 15,5625
 :r3 56,125
 :r4 64,875
 :r5 93,5625
 :r6 18050
:r2 ok
--
Převeďte desetinné číslo v kódu IEEE 754 s 8bitovým exponentem
10111101 00000000 00000000 00000000
ze dvojkové soustavy do desítkové soustavy. Jaký je správný výsledek?
 :r1 -14300
 :r2 -340
 :r3 -76,375
 :r4 -16,9375
 :r5 -0,03125
 :r6 -0,5625
:r5 ok