

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 obvyčejně nemá

: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 paměť

: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 neplatí 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 neplatí 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 neplatí?

: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 činností 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

--

Číslo 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 nepravdivé 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 nemá
: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 $0; 2^n - 1$;
- :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 $0; 2^n - 1$;
- :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  $\langle 0; 2^{n-1} \rangle$ ;
:r2  $\langle 0; 2^{n-1} \rangle$ ;
:r3  $\langle 0; 2^{n-1} \rangle$ ;
:r4  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
:r5  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
: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  $\langle 0; 2^{n-1} \rangle$ ;
:r2  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
:r3  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
:r4  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
:r5  $\langle -2^{n-1}; 2^{n-1} \rangle$ ;
: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 řídicí znaky

:r2 alfanumerické znaky

:r3 alfabetycké 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 řídicí znak s ordinální hodnotou nižší než 30

:r2 je řídicí znak s ordinální hodnotou vyšší než 30

:r3 se nevyskytuje v kódování ASCII-7

:r4 není řídicí znak

:r1 ok

--

Řídicí 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 řídicí znak neexistuje

:r2 ok

--

Řídicí 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 řídicí 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 UTF-8?

- :r1 31
- :r2 8
- :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 nejsou 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 Bipolar 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 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 vzdálenost 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 číslu 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 nepatří

- :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 sekvenční 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 $2^{\leq n}$ 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 pamět

: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 nepatří

: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 nepřičítá 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 sčítáčka 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
--

Polosčítáčka 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 udávána 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 udávána 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_{min} a U_{max}

:r3 ok

--

Pro multiplexor MUX 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 technologii 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 neplatí pro popis fyzické struktury vnitřní paměti?

- :r1 Dekodér na jeden z adresových vodičů nastaví hodnotu logická 1.
- :r2 Informace je na koncích datových vodičů zesílена 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 paměti 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 než 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 zabráňuje

:r1 použití vodičů
:r2 použití polovodičů
: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 neplatí

: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 pamětí 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 neplatí

: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 konstattní čas
:r3 záleží na nastavení 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 potřebují adresu širokou

:r1 4 bity.
:r2 8 bitů.
:r3 10 bitů.
:r4 12 bitů.
:r5 16 bitů.
:r4 ok
--

Pro adresaci 4 MB paměti potřebují 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, CMA
:r2 LDA, ADD, CMP
:r3 ADD, MOV, INR

:r1 ok

--

Příznaky pro větvení programu nikdy nemění tyto instrukce fiktivního procesoru definovaného na přednáškách

:r1 CMA, JMP, LDA
:r2 MOV, STA, JMP
:r3 STA, LDA, CMP

:r2 ok

--

Pro nastavení příznaků v procesoru definovaném na přednáškách použijí 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>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;
: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>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;
:r4 ok

```

--

```

PC -> AR, 0 -> WR, DR -> IR
PC+1 -> AR, 0 -> WR, DR -> TA<sub>L</sub>
PC+2 -> AR, 0 -> WR, DR -> TA<sub>H</sub>
TA -> AR, 0 -> WR, DR -> TAX<sub>L</sub>
TA+1 -> AR, 0 -> WR, DR -> TAX<sub>H</sub>
TAX -> AR, A -> DR, 1 -> WR
PC+3 -> 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, neefektivního 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, neefektivního 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

nesprávné?

: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 neplatí

: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 fázi každé instrukce je

- :r1 výběr operandu
- :r2 provedení instrukce
- :r3 výběr operačního znaku
- :r4 aktualizace PC

:r3 ok

--

Pro mikroinstrukci výběr operačního znaku neplatí

- :r1 cílem je vložit do instrukčního registru instrukci
- :r2 je vždy 1. fázi 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 nepatří

- :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
--

Příznaky procesoru definovaného na přednáškách mění instrukce

:r1 INR, ADD, CMA
:r2 LDA, STA
:r3 LDA, STA, JMP
:r4 LDA, STA, JMP, MOV
:r1 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 nepatří

: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 neplatí, 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 neplatí

: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 запиše 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 запиše obsah reg. A a zahájí vstupně výstupní 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 přerušeni v procesoru
- :r2 deaktivuje rutinu pro obsluhu přerušeni
- :r3 žádá o ukončení provádění procesu
- :r4 žádá o uvedeni procesoru do počátečních podmínek

:r1 ok

--

Která činnost se vykonává jako poslední při návratu z přerušeni procesoru definovaného na přednáškách?

- :r1 provedeni obslužné rutiny, která zjistí kdo žádá o přerušeni
- :r2 přerušeni provádění programu
- :r3 obnovení PC, A, ...
- :r4 úklid obsahu registrů PC, A, ...

:r3 ok

--

Pro přerušeni platí:

- :r1 přerušit lze pouze během provádění instrukce
- :r2 lze přerušit bezprostředně po zahájení obsluhy předchozího přerušeni
- :r3 o přerušeni se musí požádat signálem INTERRUPT
- :r4 přerušeni se používá typicky v kritické sekci

:r3 ok

--

Instrukce, která zakáže přerušeni 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 přerušeni

- :r1 adresa instrukce, která byla provedena před přerušeni
- :r2 adresa instrukce, která nebyla provedena v důsledku přerušeni
- :r3 adresa vrcholu zásobníku

:r2 ok

--

Během uplatnění přerušeni není provedeno

- :r1 uloženi registru PC do zásobníku
- :r2 vynulování IF
- :r3 povolení přerušeni
- :r4 uklizení registru A a dalších do zásobníku

:r3 ok

--

Která z instrukcí nepatří mezi instrukce procesoru definovaného na přednáškách, které se použijí při návratu z přerušeni

- :r1 POP
- :r2 STI
- :r3 RET
- :r4 CLI

:r4 ok

--

Co neplatí 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 nezpůsobí
: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 neplatí
: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 nenastavuje příznaky
:r1 INR
:r2 ADD
:r3 LDA
:r4 CMA
:r3 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 neplatí

- :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 neplatí

: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 neplatí

:r1 obsah špinavého rámce musím před jeho smazáním zapsat na disk

:r2 označení čistý rámec odpovídá označení rámce, 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 nesprávná 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

Jaká je maximální hodnota adresy reálné paměti v procesoru Intel 8086

```
:r1 1048575<sub>10</sub>
:r2 1048576<sub>10</sub>
:r3 FFFF<sub>16</sub>
:r4 10FFEF<sub>16</sub>
```

:r1 ok

--

Adresa 02AB:00A4₁₆ reálného režimu procesorů Intel se vyčíslí na hodnotu

```
:r1 2B54<sub>16</sub>
:r2 2CB4<sub>16</sub>
:r3 34F<sub>16</sub>
:r4 0CEB<sub>16</sub>
:r5 na žádnou z uvedených
```

:r1 ok

--

Na jaké hodnoty se nastaví bity příznakového registru provedením instrukce ADD v procesorech Intel řady 86 s operandy -5 a 8

```
:r1 CF=1, ZF=SF=OF=0
:r2 CF=ZF=SF=OF=0
:r3 CF=SF=1, ZF=OF=0
:r4 ZF=CF=OF=1, SF=0
```

:r1 ok

--

Jakou instrukcí v procesorech Intel x86 naplním registr příznaků F?

```
:r1 MOV F,AX
:r2 MOV AX,F
:r3 PUSHF
:r4 POPF
```

:r4 ok

--

Jaký je korektní postup činností při přerušení v procesoru Intel 8086?

```
:r1 do zásobníku se uloží obsah reg. příznaků
```

vynulují se příznaky IF a TF

do zásobníku se uloží CS

registr CS se naplní obsahem z n x 4+2

do zásobníku se uloží IP

registr IP se naplní obsahem z n x 4

```
:r2 vynulují se příznaky IF a TF
```

do zásobníku se uloží obsah reg. příznaků

do zásobníku se uloží CS

registr CS se naplní obsahem z n x 4+2

do zásobníku se uloží IP

registr IP se naplní obsahem z n x 4

```
:r3 Žádný z uvedených.
```

:r1 ok

--

Jaká je poslední (20bitová) adresa tabulky přerušovacích vektorů v procesoru Intel 8086 a reálných režimech procesorů vyšších

```
:r1 1023<sub>10</sub>
:r2 255<sub>10</sub>
:r3 4095<sub>10</sub>
:r4 0FFFFFFh
```

:r1 ok

--

Adresový prostor adres V/V zařízení v procesorech Intel (typicky 8086) je

```
:r1 20bitový
:r2 16bitový
```

```
:r3 8bitový
:r2 ok
--
```

Kolik různých přerušení může vzniknout v procesoru Intel 8086 a reálných režimech procesorů vyšších

```
:r1 256
:r2 128
:r3 1024
:r4 65536
:r1 ok
--
```

Která z uvedených variant instrukce MOV v procesorech Intel je nekorektní?

```
:r1 MOV prom1,AX
:r2 MOV prom1,prom2
:r3 MOV BX,prom2
:r4 MOV AX,DX
:r2 ok
--
```

Která z uvedených variant instrukce MOV v procesorech Intel je nekorektní?

```
:r1 MOV AL,BX
:r2 MOV CX,DX
:r3 MOV CL,DH
:r4 MOV BL,BL
:r1 ok
--
```

Jaké dvě různé operace se v procesorech Intel realizují jedinou instrukcí?

```
:r1 SAL a SHL provádí SHL (arit. a logický posun bitů vlevo se vždy provádí jako logický posun vlevo)
:r2 SAL a SHL provádí SAL (arit. a logický posun bitů vlevo se vždy provádí jako aritm. posun vlevo)
:r3 SAR a SHR provádí SAR (arit. a logický posun bitů vpravo se vždy provádí jako aritm. posun vpravo)
:r4 SAR a SHR provádí SHR (arit. a logický posun bitů vpravo se vždy provádí jako logický posun vpravo)
:r1 ok
--
```

Které varianty instrukce JMP v procesorech Intel přiřazují (nepřiřazují) operand do registru IP?

```
:r1 vzdálený (far) skok a nepřímý skok
:r2 blízký (near) skok a nepřímý skok
:r3 krátký (short) skok a blízký (near) skok
:r4 vzdálený (far) skok a blízký (near) skok
:r1 ok
--
```

Které varianty instrukce JMP v procesorech Intel přiřazují (nepřiřazují) operand k obsahu registru IP?

```
:r1 vzdálený (far) skok a nepřímý skok
:r2 blízký (near) skok a nepřímý skok
:r3 krátký (short) skok a blízký (near) skok
:r4 vzdálený (far) skok a blízký (near) skok
:r3 ok
--
```

Programujeme cyklus typu REPEAT, ve kterém na konci bloku testujeme, zda je hodnota $i > 5$. Pokud ano, pak provádění bloku opakujeme. Neznáme však velikost bloku, který musíme opakovat. Blok začíná návěštím "Blok" a

programujeme jej na procesoru Intel 8086. Jaká bude správná a nejbezpečnější realizace podmínky?

```
:r1 <PRE>
    CMP i,5
    JG Blok
</PRE>
:r2 <PRE>
    CMP i,5
    JLE Dále
    JMP Blok
```

Dále:

```
</PRE>
:r3 <PRE>
    CMP i,5
    JG Dále
    JMP Blok
```

Dále:

```
</PRE>
:r2 ok
--
```

Čím se procesor 8088 liší od procesoru 8086

```
:r1 8088 je určen pro vnější osmibitové prostředí
:r2 8088 je 8bitový
:r3 8088 je 16bitový
:r4 8088 je 20bitový
:r1 ok
--
```

NMI - nemaskovatelné přerušení se používá například při

```
:r1 hlášení chyb parity paměti
:r2 skocích z cyklu
:r3 přechodu do reálného režimu
:r4 žádostech o přerušení z rychlého zařízení (např. disku)
:r1 ok
--
```

Při adresaci paměti procesoru 8086 neplatí

```
:r1 používá se 20bitová adresa složená z dvou 16bitových komponent
:r2 adresu zapisujeme ve tvaru segment: offset
:r3 používá se 32bitová adresa složená z dvou 16bitových komponent
:r3 ok
--
```

Mezi segmentové registry nepatří:

```
:r1 CS
:r2 PC
:r3 SS
:r4 DS
:r2 ok
--
```

Pro registr CS platí

```
:r1 je určen pro výpočet adresy instrukce
:r2 slouží pro výpočet adresy dat
:r3 je řídicím registrem
:r4 je ekvivalentem registru PC
:r1 ok
--
```

Pro registr IP neplatí

```
:r1 je ekvivalentem registru PC
:r2 obsahuje část adresy právě prováděné instrukce
:r3 obsahuje pomocný datový segment
```

:r3 ok

--

Adresu paměti u procesoru 8086 zapisujeme ve tvaru

:r1 segment

:r2 offset

:r3 segment: offset

:r4 offset: segment

:r3 ok

--

Jakou velikost má jeden segment v procesoru 8086

:r1 16 bitů

:r2 20 KB

:r3 64 KB

:r4 1 MB

:r3 ok

--

Segment procesoru 8086 začíná na adrese dělitelné

:r1 není specifikováno, smí začínat na libovolné

:r2 16

:r3 4 K

:r4 32

:r2 ok

--

Jaká je korektní posloupnost operací při uplatnění přerušení v procesoru 8086?

:r1 IF:=0; PUSH F; PUSH CS; PUSH IP

:r2 PUSH F; IF:=0; PUSH CS; PUSH IP

:r3 PUSH AX; IF:=0; PUSH F; PUSH IP

:r4 PUSH IP; PUSH AX; PUSH F; IF:=0

:r2 ok

--

Instrukce IRET procesoru 8086 obnovuje ze zásobníku obsahy registrů

:r1 IP, AX

:r2 IP, CS

:r3 IP, CS, F

:r4 AX, CS, IP, F

:r3 ok

--

Jaký rozsah adres v procesoru 8086 bude přepsán, pokud se v nekonečné smyčce zacyklí použití instrukce PUSH AX?

:r1 00000-FFFFF

:r2 SS:0000-SS:FFFF

:r3 CS:0000-CS:FFFF

:r4 DS:0000-DS:FFFF

:r2 ok

--

V trasovacím režimu (TF=1) procesoru 8086 se provedení jedné instrukce spustí instrukcí

:r1 IRET

:r2 JMP

:r3 CALL

:r4 RET

:r1 ok

--

Trasovací režim procesoru 8086 se spouští nastavením TF=1

:r1 instrukcí SETTF

:r2 instrukcí CLTF

:r3 v příznaku TF

:r4 v registru TF
:r3 ok

--

Trasovací režim procesoru 8086 se spouští nastavením TF=1 a ukončuje se

:r1 instrukcí CLTF
:r2 instrukcí STOPT
:r3 neukončuje se
:r3 ok

--

Důvodem, proč po použití instrukce MOV SS,... v procesoru 8086 se zakazuje přerušení na dobu provádění jedné následující instrukce, je

:r1 časová náročnost instrukce MOV SS,...
:r2 kontrola přetečení obsahu zásobníku
:r3 atomické naplnění adresy vrcholu zásobníku
:r4 odstranění zbývajících návratové adresy ze zásobníku
:r3 ok

--

Nepovolená instrukce v procesoru 8086 je

:r1 MOV CS,...
:r2 MOV SS,...
:r3 MOV DS,...
:r4 MOV ES,...
:r1 ok

--

Programátor procesoru 8086 nastavuje příznaky

:r1 DF, IF, TF
:r2 OF, SF, ZF
:r3 AF, PF, CF
:r1 ok

--

Příznak ZF procesoru 8086 je nastaven

:r1 při nulovém výsledku operace
:r2 při krokovacím režimu
:r3 při aritmetickém přeplnění
:r4 při sudé paritě výsledků
:r1 ok

--

Příznak TF procesoru 8086

:r1 uvede procesor do krokovacího režimu
:r2 zabrání uplatnění vnějších maskovatelných přerušení
:r3 je nastaven při nulovém výsledku operace
:r1 ok

--

Všechny odkazy na zásobník procesoru 8086 jsou segmentovány přes registr

:r1 SS (Stack segment)
:r2 CS (Code segment)
:r3 DS (Data segment)
:r4 ES (Extra segment)
:r1 ok

--

Pro vnější přerušení procesoru 8086 neplatí

:r1 vyvolá se pomocí signálu INTERRUPT
:r2 vyvolá se např. při dělení nulou
:r3 vyvolá se pomocí signálu NMI
:r4 dělí se na maskovatelná a nemaskovatelná
:r2 ok

--

Pro vnitřní přerušení procesoru 8086 neplatí

:r1 vyvolá se chybou při běhu programu
:r2 je generováno programově
:r3 vyvolá se instrukcí INT n
:r4 je generováno řadičem přerušeni
:r4 ok
--

Akce, která se neprovádí při přerušeni procesoru 8086:

:r1 vynulují se příznaky IF a TF
:r2 provede se instrukce OUT
:r3 do zásobníku se uloží registr CS
:r4 do zásobníku se uloží registr IP
:r2 ok
--

Pro tabulku adres rutin obsluhujících přerušeni procesoru 8086

neplatí

:r1 začíná na adrese 0:0000
:r2 začíná na začátku adresového prostoru
:r3 má 256 řádků
:r4 začíná na adrese SS:0000
:r4 ok
--

Při přerušeni v procesoru 8086 se jako první operace provádí

:r1 do zásobníku se uloží registr příznaků (F)
:r2 vynulují se příznaky IF a TF
:r3 registr IP se naplní 16bitovým obsahem adresy $n \times 4$
:r4 registr CS se naplní 16bitovým obsahem adresy $n \times 4 + 2$
:r1 ok
--

Při návratu z přerušeni v procesoru 8086 se provádí instrukce IRET, pro níž neplatí

:r1 ze zásobníku se obnoví registr IP
:r2 ze zásobníku se obnoví registr CS
:r3 ze zásobníku se obnoví příznakový registr
:r4 ze zásobníku se obnoví registr SS
:r4 ok
--

Návrat do přerušeniho procesu v procesoru 8086 typicky zajistí instrukce

:r1 IRET
:r2 MOV
:r3 OUT
:r4 POP
:r1 ok
--

Mezi rezervovaná přerušeni procesoru 8086 nepatří

:r1 pokus o dělení nulou
:r2 krokovací režim
:r3 ladící bod
:r4 časovač
:r4 ok
--

Pro trasovací režim procesoru 8086 neplatí

:r1 po provedeni instrukce je generováno přerušeni INT 1
:r2 procesor je uveden do krokovacího režimu příznakem TF (Trace Flag)
:r3 krokovací režim využívá instrukci IRET
:r4 probíhá, když je TF nastaven na nulu
:r4 ok
--

Příznak TF procesoru 8086 se nastaví na jedničku

:r1 při obnově příznakového registru (F) ze zásobníku instrukcí IRET
:r2 instrukcí SETTF
:r3 při obnově registru IP ze zásobníku instrukcí IRET
:r4 žádná z uvedených možností
:r1 ok
--

Signál RESET procesoru 8086 neprovede

:r1 vynuluje IP
:r2 vynuluje příznakový registr
:r3 nastaví TF = 1
:r4 vynuluje SS
:r3 ok
--

Chci naplnit registr AH procesoru 8086 hodnotou 50, které řešení
není správné

:r1 MOV AH,50
:r2 PADESAT DB 50
MOV AH,PADESAT
:r3 MOV AH,[50]
:r3 ok
--

Chci naplnit registr AH procesoru 8086 obsahem adresy 50, které řešení je
správné

:r1 MOV AH,50
:r2 PADESAT DB 50
MOV AH,50
:r3 MOV AH,[50]
:r4 žádná z uvedených možností
:r3 ok
--

Instrukce procesoru 8086 MOV AH,[BX] provede

:r1 hodnota registru BX se uloží do registru AH
:r2 hodnota, která je na adrese v registru BX, se uloží do AH
:r3 registr BX se naplní hodnotou z adresy uložené v registru AH
:r4 hodnota registru AH se uloží do registru BX
:r2 ok
--

Instrukce procesoru 8086 MOV AH,[BX][DI] provede

:r1 hodnota vzniklá sečtením obsahů registrů BX a DI se uloží do
registru AH
:r2 hodnota, která je na adrese, jež vznikne součtem adres v registrech
BX a DI se uloží do AH
:r3 hodnota, která je uložena v AH se uloží do registrů BX a DI
:r4 hodnota, která je na adrese, jež vznikne rozdílem adres v registrech
BX a DI se uloží do AH
:r2 ok
--

Který ze zápisů instrukcí procesoru 8086 není korektní operací?

:r1 MOV AX,BX
:r2 MOV AX,[BX]
:r3 MOV AX,PROM[BX][DI]
:r4 žádná z uvedených možností
:r4 ok
--

Pro instrukci MOV procesoru 8086 neplatí

:r1 mění příznaky
:r2 nelze s ní měnit registr CS
:r3 má povolen tvar MOV BX,CX

:r4 nemá povolen tvar MOV adresa,adresa
:r1 ok
--

Který ze zápisů instrukcí procesoru 8086 je špatně

:r1 MOV CS,DS
:r2 MOV DS,adresa
:r3 MOV adresa,DS
:r4 MOV CX,DX
:r1 ok
--

Pro instrukci procesoru 8086 MOV SS,... platí

:r1 po dobu trvání následující instrukce je zakázáno přerušení
:r2 po dobu trvání předchozí instrukce bylo zakázáno přerušení
:r3 je nepovolená operace
:r1 ok
--

Pro aritmetické instrukce procesoru 8086 platí

:r1 nesmí nastavovat příznaky
:r2 nepatří sem instrukce ADD
:r3 nepatří sem instrukce INC
:r4 žádná z uvedených možností
:r4 ok
--

Pro znaménkové rozšíření procesoru 8086 neplatí

:r1 do všech bitů nové horní poloviny se zkopíruje znaménkový bit
původního objektu
:r2 znaménko je zachováno
:r3 všechny bity původního objektu se zkopírují do jeho nové horní
poloviny
:r3 ok
--

Instrukce procesoru 8086 ADC se používá

:r1 při sčítání širších objektů
:r2 při násobení dvou čísel
:r3 při odčítání s výpůjčkou
:r4 při přičítání k obsahu registru CX
:r1 ok
--

Při násobení reálných čísel procesoru 8086 použijeme instrukci

:r1 IMUL
:r2 MUL
:r3 IDIV
:r4 žádná z uvedených možností
:r4 ok
--

Který z následujících skoků procesoru 8086 mění registr CS?

:r1 vzdálený (far)
:r2 krátký (short)
:r3 blízký (near)
:r4 žádná z uvedených možností
:r1 ok
--

Který skok procesoru 8086 pracuje se 16bitovým operandem?

:r1 vzdálený (far)
:r2 krátký (short)
:r3 blízký (near)
:r4 žádná z uvedených možností
:r3 ok

--

Který skok procesoru 8086 pracuje s 8bitovým operandem?

- :r1 vzdálený (far)
- :r2 krátký (short)
- :r3 blízký (near)
- :r4 žádná z uvedených možností

:r2 ok

--

Pro podmíněný skok procesoru 8086 neplatí

- :r1 je vždy krátký
- :r2 reaguje na obsah příznaků
- :r3 vždy mění registr CS
- :r4 cílová adresa se vytvoří 8bitovým přírůstkem

:r3 ok

--

Označení typu GE, LT, ... jsou vyjádřením

- :r1 příznaků příznakového registru procesoru
- :r2 označení sběrnic procesoru definovaného na přednáškách
- :r3 porovnání velikosti čísel
- :r4 subtypu procesoru

:r3 ok

--

Do zásobníku procesoru 8086 se vkládají

- :r1 8bitové objekty
- :r2 16bitové objekty
- :r3 32bitové objekty
- :r4 64bitové objekty

:r2 ok

--

Pro instrukci POP SS,... platí

- :r1 po dobu trvání následující instrukce je zakázáno přerušení
- :r2 po dobu trvání předchozí instrukce bylo zakázáno přerušení
- :r3 je nepovolená operace

:r1 ok

--

Jaký je rozdíl mezi instrukcí RET a RETF procesoru 8086

- :r1 RETF naplní i registr CS
- :r2 RET naplní i registr CS
- :r3 RET i RETF pracují s 32bit. objekty, ale pouze RETF naplňuje registr CS
- :r4 RET i RETF pracují s 32bit. objekty, ale pouze RET naplňuje registr CS

:r1 ok

--

Instrukce HALT procesoru 8086

- :r1 vynuluje registry
- :r2 vypne počítač
- :r3 uvede procesor do stavu čekání
- :r4 vynuluje příznaky a registry

:r3 ok

--

Proč instrukce STI procesoru 8086 nepovoluje přerušení ihned?

- :r1 aby mohlo být provedeno instrukcí MOV SP,... atomické naplnění ukazatele vrcholu zásobníku
- :r2 aby mohl být nepřerušitelně zastaven procesor instrukcí HLT
- :r3 aby mohl být atomicky nepřerušitelně uložen ukazatel vrcholu zásobníku

:r4 aby byla nepřerušitelně ze zásobníku vybrána adresa přerušené instrukce
:r4 ok
--

Kde začíná segment reálného režimu (procesoru 8086)?

:r1 na libovolné adrese
:r2 na adrese dělitelné 4
:r3 na adrese dělitelné 16
:r4 na adrese dělitelné 32
:r3 ok
--

Adresa reálného režimu procesorů Intel x86 ve tvaru segment : offset
01B2:0015 představuje dvacetibitovou adresu (hexadecimálně)

:r1 01B35
:r2 01B17
:r3 011B5
:r4 002B5
:r1 ok
--

Registr IP procesoru Intel 8086 obsahuje

:r1 segmentovou část adresy právě prováděné instrukce
:r2 offsetovou část adresy právě prováděné instrukce
:r3 segmentovou část adresy následující instrukce
:r4 offsetovou část adresy následující instrukce
:r2 ok
--

Všechny odkazy na zásobník v procesoru Intel 8086 jsou segmentovány přes registr

:r1 AX
:r2 F
:r3 SS
:r4 CS
:r5 DS
:r3 ok
--

Instrukce IRET reálného režimu procesoru Intel x86 zajišťuje

:r1 návrat do přerušného procesu a jeho pokračování
:r2 přerušení procesu
:r3 přerušení procesu po provedení následující instrukce
:r4 návrat od přerušného procesu. Jeho pokračování zajistí jiná instrukce
:r5 návrat z podprogramu
:r1 ok
--

Je-li v procesoru 8086 nastaven příznak OF=1 a následně je provedena instrukce INTO, nastane

:r1 přepnutí do krokovacího režimu
:r2 přerušení INT 4
:r3 návrat do přerušného procesu
:r4 obnovení registru IP ze zásobníku
:r2 ok
--

Procesor 8086 poskytuje pro adresování V/V bran

:r1 8bitovou adresu
:r2 16bitovou adresu
:r3 20bitovou adresu
:r4 24bitovou adresu
:r2 ok

--

Která z následujících operací procesoru Intel x86 je nekorektní

:r1 MOV BX,CX
:r2 MOV DI,10000
:r3 MOV AX,CS
:r4 MOV CS,AX

:r4 ok

--

Instrukce AND Intel x86 provádí

:r1 logický součin
:r2 logický součet
:r3 přidání hodnoty ze zdrojového registru do cílového
:r4 součet dvou čísel
:r5 přičtení čísla

:r1 ok

--

Instrukce IN AX,DX procesoru x86 zajišťuje

:r1 přenos slabiky z AL do V/V brány podle DX
:r2 přenos slova z AL do V/V brány podle DX
:r3 přenos slabiky z V/V brány podle DX do registru AX
:r4 přenos slova z V/V brány podle DX do registru AX

:r4 ok

--

Instrukce INC CL provede v procesoru x86

:r1 CL := CL + CL
:r2 CL := CL + 1
:r3 CL := 1
:r4 CL := 0

:r2 ok

--

Kolika bitovou adresu při přístupu do paměti vytváří procesor Intel 8086?

:r1 16
:r2 20
:r3 24
:r4 32

:r2 ok

--

Adresa SS:SP ukazuje vždy na

:r1 dno zásobníku
:r2 na vrchol zásobníku
:r3 na adresu naposled prováděné instrukce
:r4 na adresu naposled použité V/V brány

:r2 ok

Jaká je maximální dosažitelná adresa v reálném režimu procesoru Intel 80286 a vyšších procesorů

```
:r1 0FFFFFFh
:r2 10FFFEFh
:r3 10FFFFFFh
:r4 10FFFEh
:r2 ok
--
```

Kolik řádků má tabulka popisovačů segmentů GDT nebo LDT procesoru Intel 80286 a vyšších procesorů

```
:r1 8192
:r2 4096
:r3 16384
:r4 65536
:r1 ok
--
```

Virtuální adresa procesoru Intel 80286 má celkem 30 bitů na adresaci virtuální paměti. Jak velká tato virtuální paměť může být?

```
:r1 1 GB
:r2 4 GB
:r3 2 GB
:r4 16 MB
:r1 ok
--
```

S obsahem instrukčního segmentu procesoru Intel 80286 je povoleno následující:

```
:r1 číst a provádět; mám-li potřebná práva, pak i zapisovat
:r2 pouze provádět a možná i číst; mám-li potřebná práva
:r3 cokoli, pokud mám potřebná práva
:r2 ok
--
```

Popisovač segmentu s LDT (tabulka popisovačů lokálního adresového prostoru) se v procesoru Intel 80286 smí nacházet v těchto tabulkách

```
:r1 pouze v GDT
:r2 v GDT i v LDT
:r3 v GDT a IDT
:r4 v žádné z nich
:r1 ok
--
```

Popisovač segmentu s GDT (tabulka popisovačů globálního adresového prostoru) se v procesoru Intel 80286 smí nacházet v těchto tabulkách

```
:r1 pouze v GDT
:r2 v GDT i v LDT
:r3 pouze v IDT
:r4 v žádné z nich
:r4 ok
--
```

V reálném režimu procesoru Intel 80286 nelze provést instrukci

```
:r1 LLDT (plnění registru LDTR)
:r2 LGDT (plnění registru GDTR)
:r3 LIDT (plnění registru IDTR)
:r4 LMSW (plnění registru MSW)
:r5 HLT (zastavení procesoru)
:r1 ok
--
```

Jaký je rozdíl mezi přerušením typu trap a fault v procesoru Intel 80286?

```
:r1 Fault je fatální stav, ze kterého se nelze zotavit. Z přerušení trap se zotavit lze.
```

:r2 Fault pracuje s adresou ukazující na instrukci, která přerušeni způsobila. Trap poskytuje adresu ukazující na instrukci následující.
:r3 Přerušeni typu trap je obsluhováno branou z tabulky IDT a přerušeni fault je obsluhováno branou z tabulky GDT.
:r2 ok
--

Co znamená výjimka (přerušeni) "Výpadek segmentu" v procesoru Intel 80286?

:r1 procesor při vyčíslování virtuální adresy narazil na nulovou hodnotu bitu Present
:r2 procesor při vyčíslování virtuální adresy narazil na nulovou hodnotu bitu Accessed
:r3 procesoru se nepodařilo vyčíslit reálnou adresu z virtuální
:r1 ok
--

Procesor 80286 má

:r1 16bit. datovou a 24bit. adresovou sběrnici
:r2 24bit. datovou a 20bit. adresovou sběrnici
:r3 32bit. datovou a 24bit. adresovou sběrnici
:r4 24bit. datovou a 32bit. adresovou sběrnici
:r1 ok
--

Procesor 80286 má

:r1 chráněný a reálný režim
:r2 chráněný a virtuální režim
:r3 sběrnicový a reálný režim
:r4 reálný a nereálný režim
:r1 ok
--

Pro chráněný režim procesoru 80286 neplatí

:r1 není možnost jej softwarově vypnout
:r2 tabulka vektorů přerušeni má velikost 1 KB
:r3 poskytuje prostředky 4úrovňové ochrany
:r4 adresu 16 MB reálné paměti
:r2 ok
--

Pro registr MSW procesoru 80286 neplatí

:r1 slouží k zapnutí chráněného režimu
:r2 slouží k zapnutí reálného režimu
:r3 plní se instrukcí LMSW
:r4 čte se instrukcí SMSW
:r2 ok
--

Signál RESET u procesoru 80286

:r1 zapíná chráněný režim procesoru
:r2 zapíná reálný režim procesoru
:r3 vypíná koprocessor
:r4 žádná z uvedených možností
:r2 ok
--

Bit P popisovače datového segmentu procesoru 80286 nastavený na 1 sděluje:

:r1 obsah segmentu je uložen na disku
:r2 obsah segmentu je prázdný
:r3 obsah segmentu je uložen v reálné paměti
:r4 je vždy automaticky nastaven na jedničku
:r3 ok
--

Bit ED datového segmentu procesoru 80286 určuje

- :r1 zda datový segment obsahuje zásobník
- :r2 přístupová práva k segmentu
- :r3 zakazuje čtení obsahu segmentu
- :r4 zakazuje zápis do segmentu

:r1 ok

--

Bit C (Conforming) popisovače instrukčního segmentu procesoru 80286

:r1 může způsobit změnu úrovně oprávnění pro podprogramy volané v tomto segmentu

- :r2 indikuje směr rozšiřování segmentu
- :r3 je nastaven na jedna, je-li procesor v reálném režimu
- :r4 žádná z uvedených možností

:r1 ok

--

Pro registr GDTR procesoru 80286 neplatí

- :r1 má délku 5 bajtů
- :r2 při spuštění chráněného režimu se do něj vkládá adresa tabulky GDT
- :r3 naplňuje se instrukcí LGDT
- :r4 označuje segment stavu procesoru

:r4 ok

--

Pro TSS (segment stavu procesoru 80286) neplatí

:r1 na segment TSS ukazuje popisovač systémového segmentu, který může být umístěn pouze v GDT

- :r2 slouží k uložení kontextu procesu, kterému bylo odebráno řízení
- :r3 je to ukazatel, jestli je procesor 80286 v chráněném režimu
- :r4 každý proces má vlastní TSS

:r3 ok

--

Interrupt Descriptor Table (IDT) procesoru 80286 nemá popisovač

- :r1 brána zpřístupňující TSS
- :r2 brána pro maskující přerušení
- :r3 brána pro nemaskující přerušení
- :r4 brána pro V/V operace

:r4 ok

--

Pro Interrupt Descriptor Table (IDT) procesoru 80286 neplatí

- :r1 obsahuje až 256 popisovačů rutin obsluhujících přerušení
- :r2 její adresu obsahuje IDTR
- :r3 slouží k uložení kontextu procesu, kterému bylo odebráno řízení
- :r4 obsahuje nejvýše tolik popisovačů, kolik dovoluje limit segmentu

:r3 ok

--

Kolik řádků má Interrupt Descriptor Table (IDT) procesoru 80286?

- :r1 24
- :r2 64
- :r3 128
- :r4 256
- :r5 8192

:r4 ok

--

Který z následujících názvů nespecifikuje kategorii přerušení generovanou procesorem 80286?

- :r1 Fault
- :r2 Trap
- :r3 Abort
- :r4 Flag

:r4 ok

--

Která z možností nepatří mezi rezervovaná přerušení 80286?

:r1 dělení nulou

:r2 přeplnění

:r3 chybný operační kód

:r4 výpadek systému

:r4 ok

--

Zapnutí chráněného režimu procesoru 80286 neznamená

:r1 změnu způsobu adresace

:r2 nastavení bitu PE=1 registru MSW

:r3 vypnutí reálného režimu

:r4 restart procesoru

:r4 ok

--

Počet lokálních adresových prostorů procesoru 80286 typicky se rovná

:r1 počtu procesů

:r2 počtu uplatněných přerušení

:r3 jedné

:r4 záleží na operačním systému

:r5 velikosti tabulky GDT

:r1 ok

Procesor Intel 80386 je

:r1 32bitový procesor s 32bitovou adresovou a datovou sběrnici

:r2 32bitový procesor s 24bitovou vnější a 32bitovou vnitřní adresovou sběrnici

:r3 32bitový procesor s 24bitovou adresovou a 32bitovou datovou sběrnici

:r1 ok

--

Selektor v chráněném režimu procesoru Intel 80386 je

:r1 16bitový

:r2 32bitový

:r3 48bitový

:r4 64bitový

:r1 ok

--

Procesor Intel 80386 pracuje s těmito možnými adresami

:r1 48bitovou logickou adresou, 32bitovou lineární adresou a 32bitovou fyzickou adresou

:r2 64bitovou logickou adresou, 48bitovou lineární adresou a 32bitovou fyzickou adresou

:r3 48bitovou logickou adresou, 48bitovou lineární adresou a 32bitovou fyzickou adresou

:r1 ok

--

Kolika bity plní programátor segmentové registry v procesoru Intel 80386 a vyšších typech?

:r1 16

:r2 32

:r3 48

:r4 64

:r1 ok

--

Registr EAX je použit od procesorů

:r1 286

:r2 386

:r3 486

:r4 Pentium

:r5 x86-64

:r2 ok

--

Stránkováním se v procesoru Intel 80386 transformuje

:r1 logická adresa na lineární

:r2 fyzická adresa na lineární

:r3 lineární adresa na fyzickou

:r4 logická adresa na fyzickou

:r3 ok

--

Největší možná velikost segmentu v procesoru Intel 80386 a vyšších typech je

:r1 64 KB

:r2 1 MB

:r3 4 MB

:r4 1 GB

:r5 4 GB

:r5 ok

--

Velikost stránky v procesoru Intel 80386 a vyšších typech je

:r1 maximálně 4 KB

:r2 právě 4 KB

:r3 maximálně 1 KB
:r4 právě 1 KB
:r2 ok

--

Co znamená "Mapa přístupných V/V bran" v procesoru Intel 80386?

:r1 Seznam existujících V/V adres na počítači.
:r2 Seznam V/V adres dostupných jednomu konkrétnímu (typicky V86) procesu.
:r3 Seznam V/V adres dostupných (typicky V86) procesům chráněného režimu.
:r2 ok

--

Jaká část adresy vstupující do stránkovací jednotky není stránkováním postihnuta (v procesoru Intel 80386)?

:r1 dolních 12 bitů
:r2 dolních 10 bitů
:r3 horních 10 bitů
:r4 horních 20 bitů
:r1 ok

--

Kolik bitů je nezbytných pro uložení adresy stránkovací tabulky (zpravidla ve stránkovacím adresáři) a stránkovacího adresáře (zpravidla v CR3)?

:r1 20
:r2 32
:r3 16
:r1 ok

--

Pro procesor 80386 neplatí

:r1 datová sběrnice má 32 bitů
:r2 lze použít stránkování
:r3 data se do/z paměti přenášejí po 4 bajtech
:r4 adresová sběrnice má 24 bitů
:r4 ok

--

Pro adresaci v chráněném režimu procesoru 80386 neplatí

:r1 offset je 16bitový
:r2 selektor je stejný jako v 80286
:r3 báze segmentu je 32bitová
:r4 limit segmentu může být až 4 GB
:r1 ok

--

Instrukční prefix je realizován

:r1 jedním bajtem uloženým před prvním bajtem operačního kódu
:r2 jedním bitem v operačním kódu
:r3 n-bitovým vzorkem v operačním kódu
:r1 ok

--

Co znamená, že stránkovací jednotka procesoru 80386 není zapnuta

:r1 fyzická adresa je totožná s lineární adresou
:r2 fyzická adresa obsahuje 48 bitů
:r3 lineární adresa obsahuje 48 bitů
:r4 lineární adresa je totožná s logickou adresou
:r1 ok

--

Pro stránkování procesoru 80386 platí

:r1 je povinně zapnuto
:r2 je-li vypnuto, tak se lineární adresa transformuje na logickou

:r3 je-li zapnuto, tak se lineární adresa transformuje na fyzickou
:r4 žádná z uvedených možností
:r3 ok

--

Pro stránkový adresář procesoru 80386 neplatí

:r1 má velikost právě jedné stránky
:r2 ukazuje na max. 1024 stránkových tabulek
:r3 je k dispozici pouze se zapnutým stránkováním
:r4 žádná z uvedených možností
:r4 ok

--

Pro bit D (Dirty) při stránkování procesoru 80386 neplatí

:r1 procesor ho nastaví na jedničku při zápisu do rámce
:r2 procesor jej nenuluje, to má na starost software
:r3 rozlišuje, jestli je rámec špinavý nebo čistý
:r4 pokud je nastaven na jedničku, tak je rámec vybrán za oběť
:r4 ok

--

Pro TLB neplatí

:r1 funguje na principu asociativní paměti
:r2 je zapnuto pouze v chráněném režimu procesoru 80286
:r3 je to vyrovnávací paměť
:r4 při vyprazdňování se vynulují bity V (validita)
:r2 ok

--

Procesor 80386 má 32 bitovou adresovou sběrnici A2 až A31, což znamená,
že

:r1 do paměti se jde alespoň pro 1 bajt
:r2 do paměti se jde alespoň pro 2 bajty
:r3 do paměti se jde alespoň pro 4 bajty
:r4 do paměti se jde alespoň pro 8 bajtů
:r3 ok

Jaký má význam interní vyrovnávací paměť v procesoru Intel 80486?

:r1 Pamatuje si posledních několik transformovaných lineárních adres na fyzické.

:r2 Pamatuje si několik posledních obsahů adres čtených z fyzické paměti vč. okolí.

:r3 Vyrovnává rozdíly toku dat mezi interními jednotkami procesoru pro proudové zpracování (pipeline).

:r2 ok

--

Kolik bitů by potřeboval algoritmus LRU v interní vyrovnávací paměti procesoru Intel 80486 k tomu, aby úplně fungoval pro výběr ze čtyř položek na řádku (předpokládejme, že by byl realizován neúplnou maticí)?

:r1 3

:r2 4

:r3 6

:r4 8

:r5 10

:r3 ok

--

Procesor 80486 nemá

:r1 datovou sběrnici 32bitů

:r2 adresovou sběrnici 32bitů

:r3 integrovaný matematický koprocessor

:r4 žádná z uvedených možností

:r4 ok

--

Procesor 80486 se od procesoru 80386 neliší v

:r1 velikosti vnějších sběrnic

:r2 interní vyrovnávací paměti

:r3 nové technologii, která se blíží k RISCovým procesorům

:r4 jednotce operací v pohyblivé řádové čárce

:r1 ok

--

V procesoru Pentium

:r1 se dynamicky předvídá výskyt nepodmíněné skokové instrukce

:r2 se dynamicky předvídá výskyt podmíněné skokové instrukce

:r3 se dynamicky předvídá výsledek vyhodnocení podmínky podmíněné skokové instrukce

:r4 se staticky předvídají nepodmíněné skoky dvouvariantními instrukcemi

:r3 ok

--

Autorem architektury x86-64 je firma

:r1 AMD

:r2 Intel

:r3 Hewlett-Packard

:r1 ok

--

Reálný režim se používá u procesorů

:r1 286

:r2 386

:r3 486

:r4 x86-64

:r5 všech vyjmenovaných

:r5 ok

--

V 64bitovém režimu architektury x86-64 se potlačuje

:r1 stránkování

:r2 segmentace

:r2 ok

--

Zástupcem architektury x86-64 je procesor

:r1 Itanium

:r2 Opteron

:r2 ok

--

Jakou velikost má selektor v 64bitovém režimu architektury x86-64?

:r1 16 bitů

:r2 32 bitů

:r3 64 bitů

:r1 ok

Který rys je vlastní technologii procesorů RISC?

:r1 usnadnění programování pro člověka programátora

:r2 zrychlení provádění poskytnutím co nejbohatších instrukcí

:r3 integrování vnější paměti dovnitř procesoru

:r4 poskytnutí velkého počtu registrů v procesoru

:r4 ok

--

Který procesor má více různých instrukcí pracujících s operandy v paměti?

:r1 RISC

:r2 CISC

:r2 ok

Základní šířka dat interně zpracovávaných koprocесorem pro výpočty v pohyblivé řádové čárce je

:r1 80 bitů
:r2 128 bitů
:r3 64 bitů
:r4 40 bitů
:r5 32 bitů

:r1 ok

--

Nejmenší záporné číslo (největší v absolutní hodnotě; číslo na levé hranici rozsahu zobrazení) v IEEE 754 má

:r1 znaménko mantisy 1, největší kladné zobrazitelné číslo v exponentu.
:r2 znaménko mantisy 1, nejmenší záporné zobrazitelné číslo v exponentu.
:r3 znaménko mantisy 1, nulový exponent.
:r4 znaménko mantisy 0.

:r1 ok

Signály STROBE a BUSY používá rozhraní

:r1 RS-232
:r2 V.24
:r3 Centronics
:r4 USB
:r3 ok

--

Paralelní rozhraní je

:r1 RS-232.
:r2 Centronics.
:r2 ok

--

Rozhraní Centronics: Signál !STROBE je v aktivní úrovni

:r1 dokud neuplyne doba "předstih"
:r2 dokud neuplyne doba "přesah"
:r3 dokud tiskárna signálem BUSY neoznámí konec zpracování
:r4 pevně stanovenou dobu
:r4 ok

--

Rozhraní Centronics: Signál !STROBE je v aktivní úrovni

:r1 když přenáší hodnotu logická "0"
:r2 když přenáší hodnotu logická "1"
:r1 ok

--

Rozhraní RS-232C: Přenos dat tímto rozhraním je:

:r1 synchronní
:r2 asynchronní
:r3 synchronní i asynchronní
:r4 nic z toho
:r3 ok

--

Rozhraní RS-232C: Jaké zapojení nulmodemu je nesmyslné?

:r1 SG--SG, TxD--RxD, RxD--TxD
:r2 SG--SG, TxD--TxD, RxD--RxD
:r3 SG--SG, TxD--RxD, RxD--TxD, RTS+CTS--DCD, DCD--RTS+CTS
:r2 ok

--

USB při komunikaci používá protokol

:r1 Master-Slave
:r2 CSMA/CD
:r3 Token-Ring
:r1 ok