1. Základní vlastnosti číslicového počítače:
   Nespojité – diskrétní (digitální, číslicové) zobrazení dat

2. Mooreův zákon o vývoji hustoty integrace

Procesory: Logická kapacita: o 30% za rok, Hodinová frekvence: o 20% za rok; Hlavní paměť: DRAM kapacita: o 60% za rok (4x každé 3 roky), Rychlost – přístupová doba: o 10% za rok, Cena za bit: snížení o 25% za rok, Disk: Kapacita: o 60% za rok, Využití dat: o 100% každých 9 měsíců, Počítačové sítě: Sířka pásma o 100% za rok!

3. Typy reprezentace systému a úrovně abstrakce

Funkční (behavioral or functional representation) = Co to má dělat, Popis funkce ne implementace, Black-box + závislosti výstupů na vstupech v čase, Strukturní = Jak je to zapojeno, Popis implementace bez zvláštního popisu fce.(ta vyplývá ze vzájemného spojení bloků o známé fci), Vnitřek black-boxů, Fyzikální = Jak to vyrobit, Popisuje fyzikální vlastnosti každého black-boxu, Popisuje přesné vztahy mezi bloky (velikost, hmotnost, spotřebu, zahřátí,a to v každém bodě, vstupním i výstupním pinu)

4. Postup návrhu počítače metodou „zdola nahoru“
   Jedná se o návrh, kde se začíná hradly a vnitřní strukturou, pokračuje přes assembler a řešení problémů, jak ukládat data, jak vše postavit, a končí psaním algoritmů

5. Členění počítačového software
   Firmware, Operační systém, Vývojářský software, Aplikace

6. Členění počítačového hardware
   Architektura procesoru, Paměťová hierarchie, Systémová rozhranní, Uživatelská rozhraní

7. Architektura počítače – základní bloky
   Procesor(výpočetní část, mezipaměť výsledků, Controller); **Paměť programu** dat(Program memory, data memory); Vstupní a výstupní zařízení (Input, Output)

8. Architektura „von Neumann“ – vlastnosti

• Instrukce a data jsou uložena v téže paměti.
• Instrukce a data nelze přenášet po sběrnici současně (společná cesta) – pomalejší činnost
• Jednodušší propojení procesoru s pamětí
• Paměť je organizována lineárně (tzn. jednorozměrně) a je rozdělena na stejně velké buňky, které se adresují celými čísly (zprav. 0, 1, 2, 3, . . .).
• Data ani instrukce nejsou explicitně označeny.
• Explicitně nejsou označeny ani různé datové typy.
• Pro reprezentaci dat i instrukcí se používají dvojkové signály.
• Instrukce se provádějí jednotlivě, a to v pořadí, v němž jsou zapsány v paměti, pokud není toto pořadí změněno speciálními instrukcemi (nazývanými skoky).
• Počítač tvoří: Processor (ALU – aritmetická a logická jednotky, registry, řídicí část (řadič – controller), hlavní paměť (main memory) – společně instrukce a data, systém přerušení, vstupní a výstupní zařízení (periferie – peripherals)

9. Architektura „Harvard“ – vlastnosti

• Instrukce a data jsou uložena v oddělených pamětech.
• Instrukce lze číst současně s přenosem dat (oddělené cesty) – rychlejší činnost
• Šířka přenosové cesty instrukcí může být jiná (větší) než přenosová cesta dat.
• Složitější propojení procesoru s pamětmi (vs „von Neumann“)
• Data a instrukce jsou explicitně označeny

• Paměť je organizována lineárně (tzn. jednorozměrně) a je rozdělena na stejně velké buňky, které se adresují celými čísly (zprav. 0, 1, 2, 3, . . .).
• Pro reprezentaci dat i instrukcí se používají dvojkové signály.
• Instrukce se provádějí jednotlivě, a to v pořadí, v němž jsou zapsány v paměti, pokud není toto pořadí změněno speciálními instrukcemi (nazývanými skoky).
• Počítač tvoří: Processor (ALU – aritmetická a logická jednotky, registry, řídicí část (řadič – controller), paměť programu (instrukcí), paměť dat, systém přerušení, vstupní a výstupní zařízení (periferie – peripherals)

10.Vývoj software – úrovně abstrakce
Řídící signály; PROCESSOR(HW); Strojový kód; ASSEMBLER(SW); Jazyk symbolických instrukcí; PŘEKLADAČ(SW); Vyšší programovací jazyk

11.Organizace hlavní paměti, adresy, obsah adresy, word, byte, bit
Hlavní paměť je rozdělena na paměťová místa, kterým jsou přiřazena nezáporná čísla (adresy). Obsah místa je „slovo“. Velikost slova v závislosti na procesoru (např. 16b, 24b, 32b, 64b), Obsah paměťového místa je na adrese adr nebo <adr>.

12.Big-endian, Little-endian
Různá organizace paměti - jeden se ukládá zepředu, druhý zezadu
Př: Chci uložit 1234ABCD

	1. způsob	2. způsob
Adresa	Big-endian	Little-endian
8001	12	CD
8002	34	AB
8003	AB	43
8004	CD	12

13.Zobrazení dat v počítači
V pevné řádové čárce(celá, racionální); v pohyblivé řádové čárce(racionální, double, float,…)
Různé soustavy: Dvojková, Šestnáctková, Desítková; Bez znaménka, se znaménkem; různě dlouhá

14.Pojem „hradlo“ a základní funkce hradel, značení hradel
Hradlo je základní stavební prvek logických obvodů, který vyčísluje logickou funkci. Typicky má jeden či více vstupů a jediný výstup. Slouží k základním logickým operacím. And, or, not, nand, nor, xor. Značení: Kolečko na výstupu značí negaci; And – půlka viagry; Not=inv – trojúhelník s kolečkem; OR – vesmírná loď; XOR – vesmírná loď s pohonem

15.Pojem hradlové pole a návrhový proces v ISEWebpack
16.Pojem logický kombinační obvod (LKO), základní vlastnosti
Je popsán logickou funkcí, vstupy mají 0 nebo 1, výstupy jsou určené pouze vstupy ve stejném okamžiku, nepamatuje si minulý stav

17.Pojem logická kombinační funkce
Více LKO dohromady.

18.Fáze návrhu číslicového systému
Specifikace, Určení vstupů a výstupů, Pravdivostní tabulky, Booleovské rovnice, Minimalizace, Návrh realizace hradel, Logická simulace, Realizace číslicového obvodu, Ověření návrhu

19.Booleova algebra, logické proměnné, logické operace, axiomy a zákony

Konečná množina prvků obsahující logické proměnné(a,b,c), binární operace(and*,or+),unární operaci negace(not) a dva logické stavy (0,1)
Axiomy – nedokazujeme - vypsat and(*) a or(+)
zákony:

$a+b = b+a$	$a.b = b.a$	Komutativní
$(a+b)+c = a+(b+c)$	$(a.b).c = a.(b.c)$	Asociativní
$a.(b+c) = a.b+a.c$	$a+(b.c) = (a+b)(a+c)$	Distributivní
$a+a = a$	$a.a = a$	Idempotentnost
$a+a´= 1$	$a.a´ = 0$	Komplentarita
$1+a = 1$	$0.a = 0$	Agresivnost
$+a = a$	$1.a = a$	Neutrálnost
$a+(a.b) = a$	$a.(a+b) = a$	Absorbce
$a+a´.b = a+b$	$a.(a´+b) = a.b$	Absorbce negace
$a´´=a$		Involuce
$(a+b)´= a´*b´$	$(a*b)´= a´+b´$	de Morganův
*Změna všech * na + a obráceně(né vždy)*		Absorbce consensu
0->1	+-> .	(OR -> AND)
1->0	. -> +	(AND -> OR)

20. Pravdivostní tabulka
Easy – And, Nand, Or,Nor, Not, Xor

21. Funkce základních hradel (dvouvstupových i vícevstupových), and, or, xor, not,…
Větší pravdivostní tabulky – XOR má lichou paritu u třech vstupů

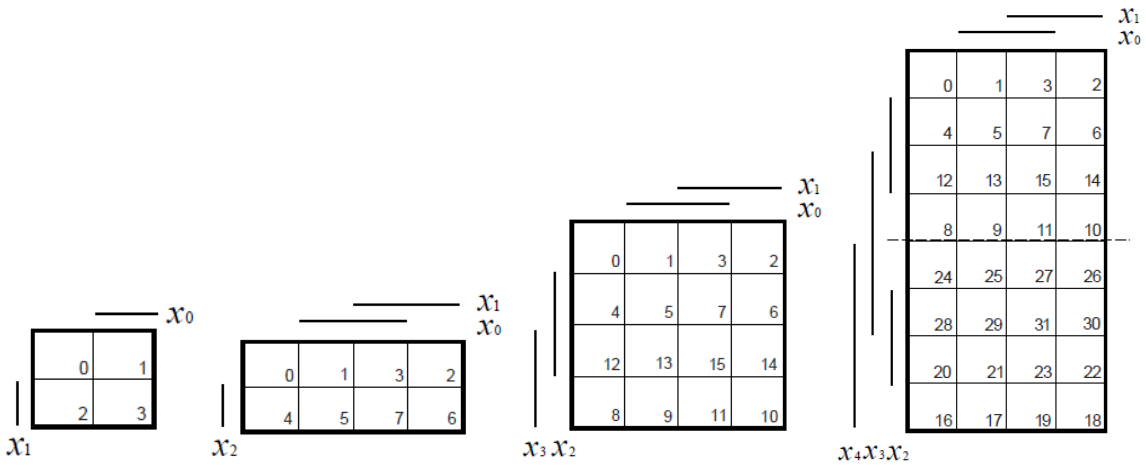
22. Obecné kombinační hradlo, funkce, zpoždění, zatížení vstupů, …
Určeno: funkčním chováním, zpožděním, zatížením, úrovněmi 0,1 na vstupu, spotřebou (nejrychlejší jsou invertory)

23. Minterm, maxterm
Minterm určuje, kde jsou jedničky, maxterm určuje, kde jsou nuly.

24. SoP (ÚNDF), PoS (ÚNKF)
Jenom odlišné zápisy:
SoP: $\Sigma m(1, 2, 3, 6)$; $f(c,b,a)= c´.b´.a+c´.b.a´+c´.b.a+c.b.a´$
PoS: $f=\Pi M_{j(0)} =\Pi M(0, 4, 5, 7)$; $f(c,b,a)=(c+b+a).(c´+b+a).(c´+b+a´).(c´+b´+a´)$

25. Minimalizace logických funkcí úpravou logického výrazu
Napíšu si tabulku, vypíšu mintermy, nebo maxtermy a potom upravuju na základě booleovské algebry.

26. Minimalizace logických funkcí pomocí K-mapy – zásady a použití
Vypsat si z tabulky mintermy, 1napsat si správnou K-mapu, vyznačit 2ntice sousedících hodnot (co největší) dle pravidel a vypisovat fce. K-mapa:



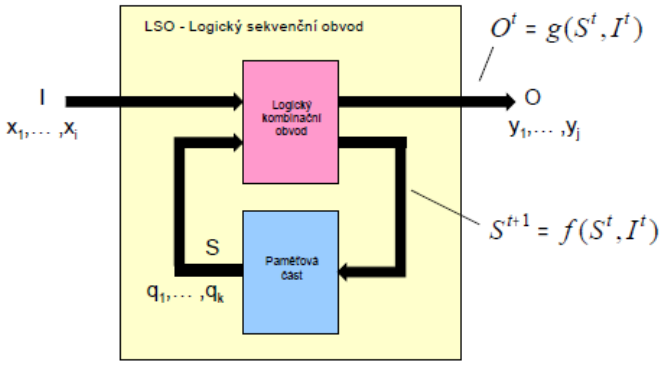
27. Realizace logické funkce z hradel
Pokud neguju celou závorku, tak se mi negují členy a zároveň mění AND na OR a obráceně

28. Rozdíl mezi kombinačním a sekvenčním obvodem
Sekvenční obvod se chová v určitých stavech jinak(závisí na posloupnosti), protože si pamatuje předchozí stav. Logický generuje výstup jen z aktuálních signálů na vstupech.

29. Pojem logický sekvenční obvod (LSO), základní vlastnosti
Obvod, který je popsán stavovým diagramem, rovnicemi, tabulkami, HDL jazykem, výstupní hodnoty závisí na aktuálním stavu vstupních proměnných a vstupních proměnných v minulém stavu (obvod má vnitřní paměť)

30. Konečný automat, přechodová a výstupní funkce
FSA, FSM, má konečný počet vnitřních, vstupních a výstupních stavů, dané kombinace se značí I_i, S_k, O_j (Input, State, Output)
Přechodová fce: $S^{t+1} = f(S^t, I^t)$ (vnitřní následující, vnitřní a vstupní současný stav)
Výstupní fce: $O^t = g(S^t, I^t)$ (výstupní, vnitřní a vstupní současný stav)

31. Obecný model logického sekvenčního obvodu (Hoffmann)



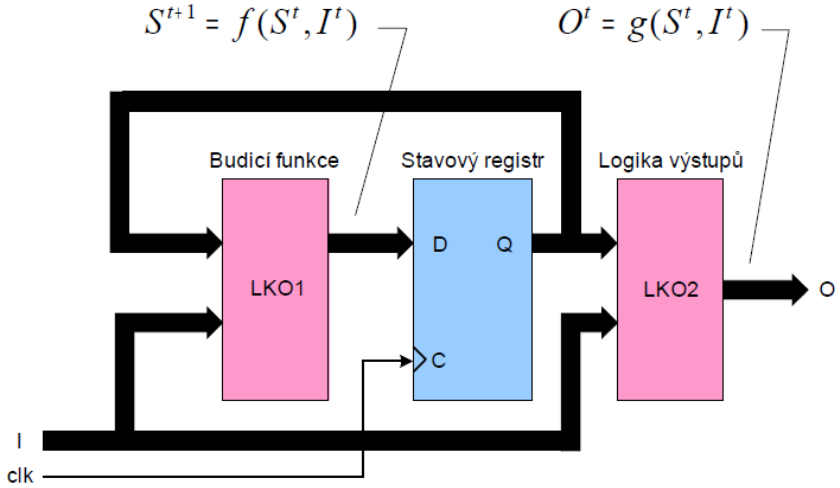
32. Rozdělení LSO (Mealy, Moore, autonomní), (synchronní, asynchronní)
Podle časové reakce na změnu vstupních proměnných:
• Asynchronní – změna stavu LSO po změně vstupních proměnných ihned (resp. s malým zpožděním v důsledku reakce vnitřních obvodů LSO).

• Synchronní – změna stavu LSO synchronizována vnějšími synchronizačními impulsy (tzv. hodiny, clock). LSO jsou navrhovány většinou jako synchronní – snadnější kontrola vnitřních signálů LSO.

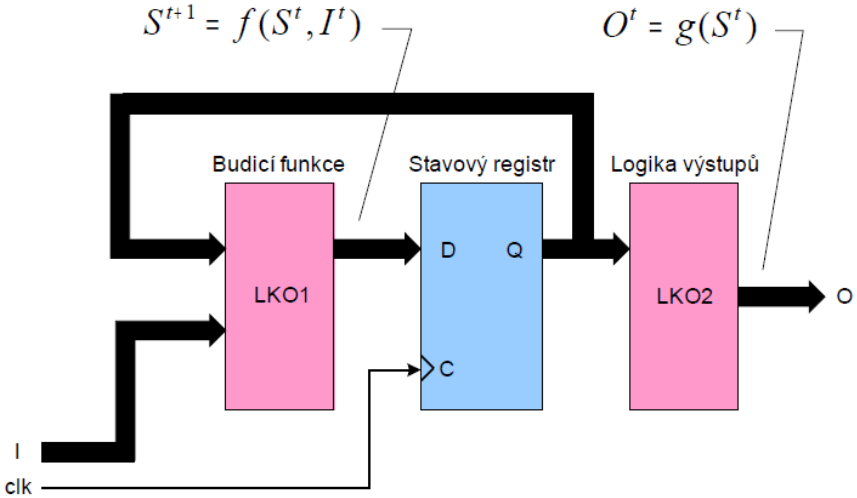
Podle způsobu výpočtu přechodové a výstupní funkce

• Automat typu Mealy: $S^{t+1} = f(S^t, I^t)$ $O^t = g(S^t, I^t)$
• Automat typu Moore: $S^{t+1} = f(S^t, I^t)$ $O^t = g(S^t)$
• Autonomní automat: $S^{t+1} = f(S^t)$ $O^t = g(S^t)$

33.Blokové schéma synchronního konečného automatu – FSA – Mealy



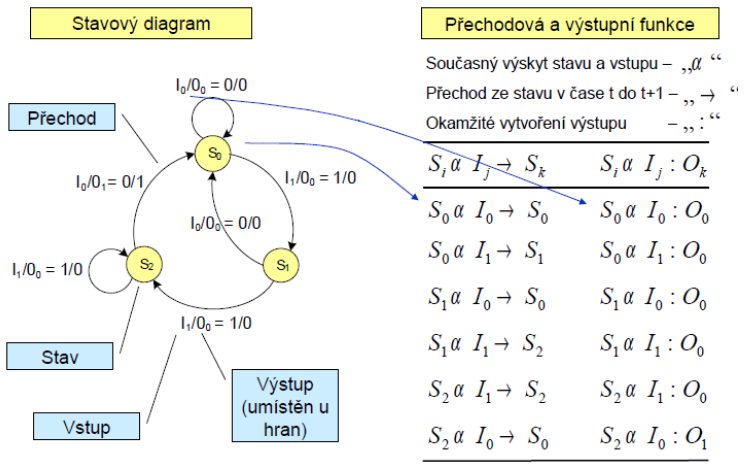
34.Blokové schéma synchronního konečného automatu – FSA – Moore



35.Blokové schéma autonomního synchronního automatu
Viz 34. Moore, ale BEZ fce „i“.

36.Formy popisu LSO
Stavový diagram(takovej ten hezkej obrázek dokola), Soustavou rovnic(S_0 a I_0 -> S_0,S_0 a I_1->S_1,…), tabulky přechodů a výstupů – obdoba rovnic, HDL – programovací jazyk.

37.Stavový diagram pro FSA Mealy, konstrukce a vlastnosti, Tabulky přechodů a výstupů, Přechodová a výstupní funkce



Tabulka přechodů

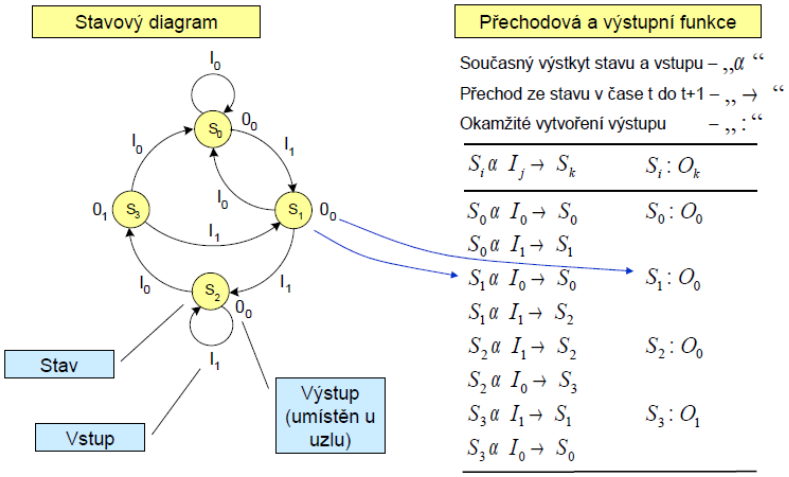
S_i	I_0	I_1
S_0	S_0	S_1
S_1	S_0	S_2
S_2	S_0	S_2

Tabulka výstupů

S_i	I_0	I_1
S_0	O_0	O_0
S_1	O_0	O_0
S_2	O_1	O_0

Výstup se projeví ihned.

38. Stavový diagram pro FSA Moore, konstrukce a vlastnosti, Tabulky přechodů a výstupů, Přechodová a výstupní funkce



Tabulka přechodů

S_i	I_0	I_1
S_0	S_0	S_1
S_1	S_0	S_2
S_2	S_3	S_2
S_3	S_0	S_1

Tabulka výstupů

S_i	O_i
S_0	O_0
S_1	O_0
S_2	O_0
S_3	O_1

Na změnu výstupu se čeká až do „tiku“ hodin.

39. Postup návrhu LSO (návrh schématu)

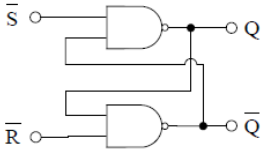
- Formulace zadání – slovní popis
- Stavový diagram (orientovaný graf přechodů a výstupů)
- Tabulky přechodů a výstupů
- Kódování vnitřních stavů a výstupů
- Zakódované tabulky přechodů a výstupů
- Budící funkce a funkce výstupů
- Minimalizace budící funkce a funkce výstupů (K–mapy)
- Návrh z hradel (z požadovaných typů) – schema
- [Logická simulace]
- Realizace z hradel
- [Časová simulace po realizaci (umístění do hradlového pole)]
- Výpočet (ověření) maximální povolené frekvence synchronizačního signálu (následující přednáška)
- Ověření v aplikaci

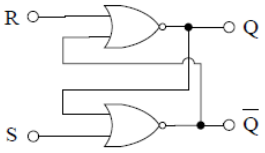
40. Postup návrhu LSO (návrh v HDL)

- Návrh obvodového řešení zápisem v HDL (VHDL, Verilog)

- Formulace zadání – slovní popis
- Stavový diagram (orientovaný graf přechodů)
- Zápis programu v HDL (Hardware Description Language)
- Syntéza zapojení (překlad programu v HDL)
- [Logická simulace]
- Realizace z hradel (z prostředků hradlového pole)(Place & Route)
- [Časová simulace po realizaci (umístění do hradlového pole)]
- Výpočet (ověření) maximální povolené frekvence synchronizačního signálu
- Ověření v aplikaci

41.Paměťové členy R-S (Latch)





R–S Latch (NAND)

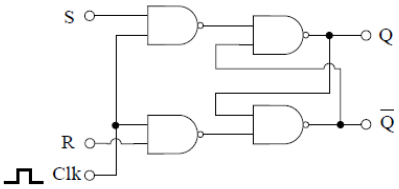
R^i	S^i	Q^{i+1}
0	0	!!!
0	1	1
1	0	0
1	1	Q^i

R–S Latch (NOR)

R^i	S^i	Q^{i+1}
0	0	Q^i
0	1	1
1	0	0
1	1	!!!

!!! – zakázaný stav

R-S (Latch, clk)



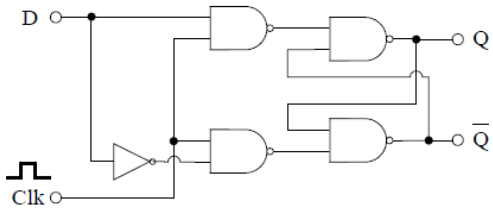
R–S Latch (Clock enable)

R^i	S^i	Clk	Q^{i+1}
0	0	1	Q^i
0	1	1	1
1	0	1	0
1	1	1	!!!
X	X	0	Q^i

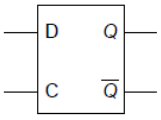
X – nezáleží

!!! – zakázaný stav

D (Latch, clk)



Symbol

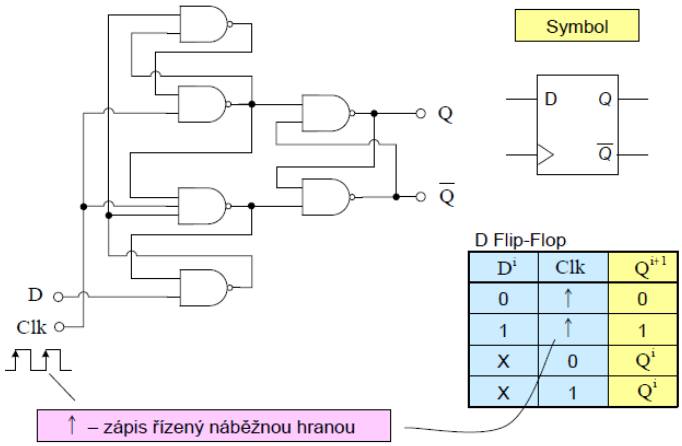


D Latch (Clock enable)

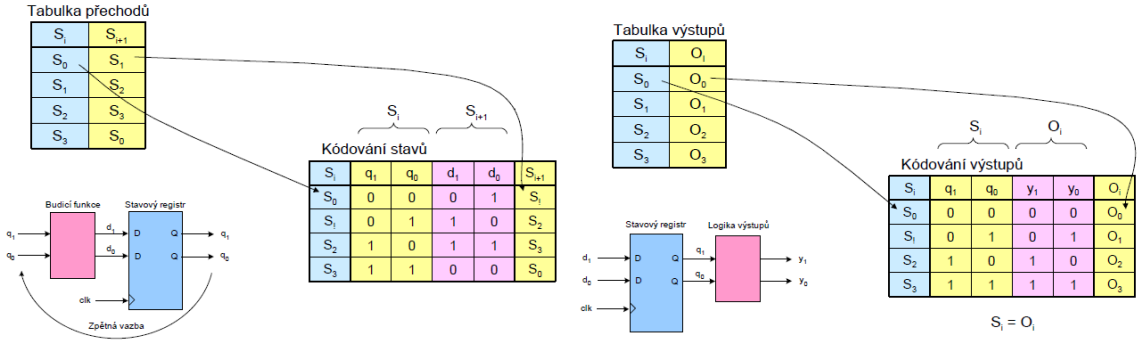
D^i	Clk	Q^{i+1}
0	1	0
1	1	1
X	0	Q^i

X – nezáleží

D (Flip-Flop, clk)



42.Kódování tabulek přechodů a výstupů



43.Stavový registr, budící funkce, logika výstupů

Viz 35.

44.Rozdíl mezi FSA Mealy a Moore

Mealy zobrazuje změnu stavu okamžitým posláním na výstup. Moore až při „tiku“ hodin.

45.Časování synchronního LSO, čím je ovlivněno

- Technologií
- Typy hradel
- Počtem vstupů u hradel
- Zatížením výstupů hradel (větvením)
- Typem klopných obvodů
- Délkou propojovacích vodičů (na plošném spoji,…)
- Vzájemnou polohou vodičů (kvalita návrhu plošného spoje)
- Rozmístěním součástek
- Počtem zemnících a napájecích vrstev
- Způsobem rozvodu napájení
- Rozmístěním blokovacích kondezátorů
- Dalšími vlivy …..

46.Časování klopného obvodu, předstih, přesah, zpoždění od hrany hodin na výstup

Předstih (Setup Time) – Vstup D musí být stabilní (ustálený) před aktivní (zde náběžnou) hranou hodinového signálu

Přesah (Hold Time) – Vstup D musí zůstat stabilní (ustálený) po aktivní (zde náběžné) hraně hodinového signálu

Zpoždění (Clock-to-Q Tieme) výstupu Q po aktivní (zde náběžné) hraně hodinového signálu

47. Stanovení maximální hodinové frekvence, kritická cesta

Všechny klopné obvody jsou řízeny stejným hodinovým signálem, všechny logické bloky jsou aktualizovány při každém taktu hodin, všechny výstupy musí být stabilní před dalším taktem. Kritická cesta: nejpomalejší cesta mezi libovolným z registrů (klop. Obvodů) Minimální perioda: funkce kritické cesty

Perioda: musí být větší než kritická cesta

48. Dekodéry

Majoritní dekodér: Nabývá hodnoty 1, když je většina vstupů rovna 1; Prioritní enkodér:Kóduje stav **n** vstupů do určeného kódu (např. binárního), na výstupu (vždy pošle nejvyšší prioritu) – použití v počítači na interrupt; Dekodér/Demultiplexer: Dekóduje kód na vstupu na kód **1** z **n** na výstupu. Použití: dekodér adresových bloků v PC. Převodníky kódu: např. Binární na Grayův od (sousední kombinace se liší pouze v jednom bitu)

49. Multiplexery

n-vstupový přepínač z **n**-vstupů na jeden výstup

50. Komparátory, jednobitový, vícebitový

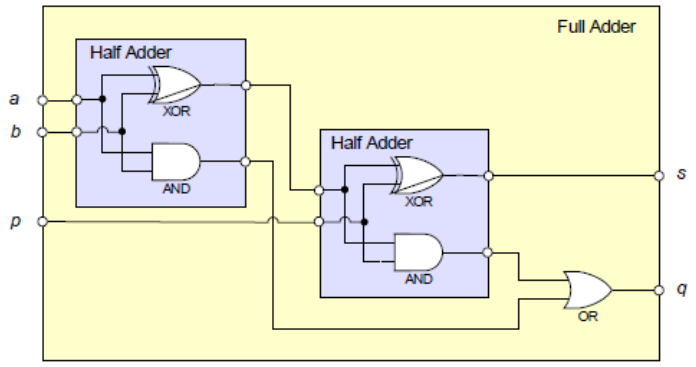
Porovnává velikost čísel, lze spojovat dohromady a porovnávat více čísel, možnosti co nenastanou, si doplňuji libovolně.

51. Shifter, Barrel shifter

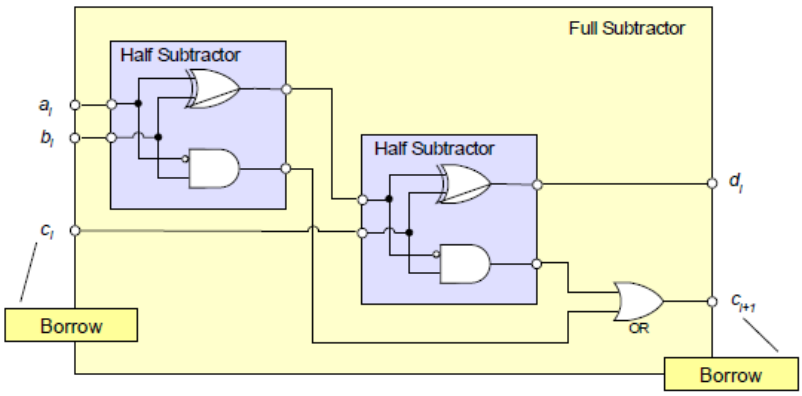
Slouží k posunu binárního řádu o jeden bit

52. Půlsčítačka, úplná sčítačka, jednobitová, vícebitová

Půlsčítačka slouží ke sčítání, ale neuvažuje přenos => sčítačka je i s přenosem

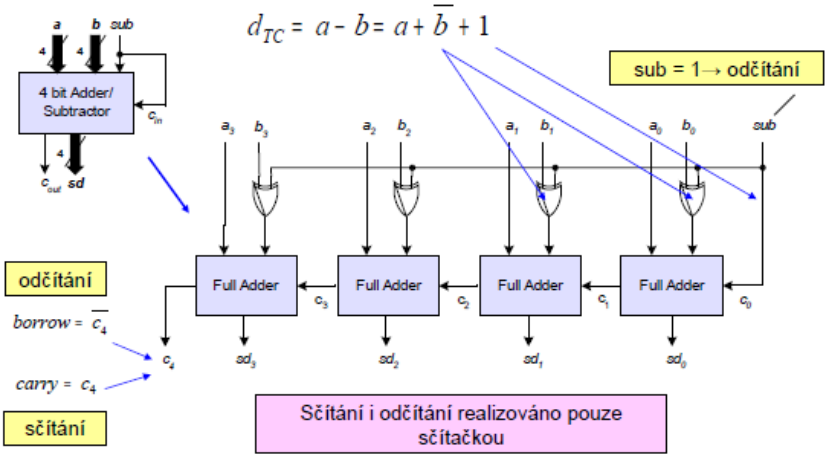


53. Půlodčítačka, úplná odčítačka, jednobitová, vícebitová



54. Sčítačka/odčítačka (ve dvojkovém doplňku)

Pro čísla kódovaná v dvojkovém doplňku platí: $d_{TC} = a - b = a + b' + 1$, sčítání i odčítání realizováno pouze sčítačkou.



55.Sekvenční bloky – přehled

56.Synchronní čítač Moore a Mealy, porovnání

Mealy zobrazuje změnu stavu okamžitým posláním na výstup. Moore až při „tiku“ hodin.

57.Vícebitový synchronní čítač

58.Posuvný registr (sériově-paralelní)

Převod sériové informace na paralelní, sériová komunikace - příjem

59.Posuvný registr (paralelně-sériový)

Převod paralelní informace na sériovou, sériová komunikace – vysílání

60.Definované časové zpoždění

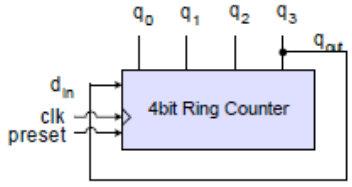
Doba, za kterou se na výstupu projeví změna. Udává se v počtu „tiků“ hodin.

61.Převod hladinového signálu na impuls

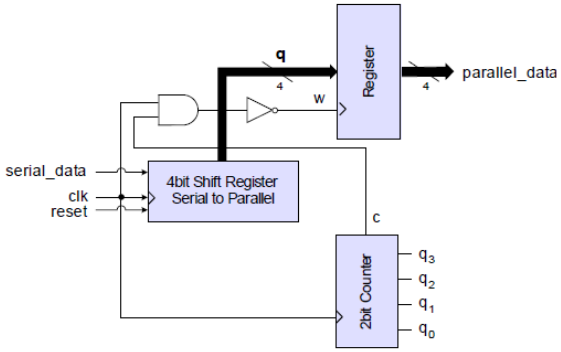
62.Obvod pro potlačení zakmitávání mechanického spínače

63.Posuvný registr s blokováním

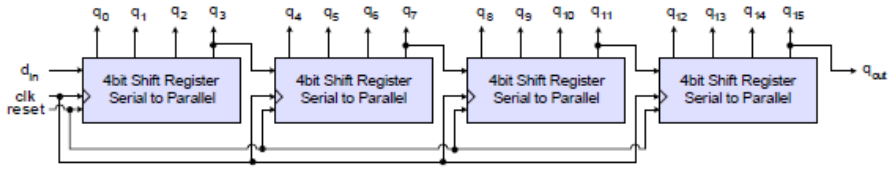
64.Kruhový čítač, vícefázové hodiny



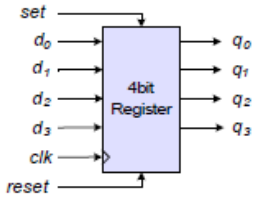
65.Převod sériové informace na paralelní – princip



66.Vícebitový posuvný registr



67.N-bitový registr řízený hodinovým signálem
n – klopných obvodů řízených společným hodinovým signálem



68.Řízení zápisu do registru s trvale běžícími hodinami (parallel load)
n – klopných obvodů řízených společným hodinovým signálem, Zápis do registru i při trvale běžících hodinách signálem load = 1

69.Typy výstupů logických členů (dvoustavový, otevřený kolektor, třístavový) – princip
Standardní výstup – Totem-pole output (Push-Pull), Dvoustavový výstup, Na výstupu vždy hodnota 0 nebo 1, Výstupy nelze navzájem spojovat; Otevřený kolektor – Open-collector output (OC), Na výstupu pouze spodní spínač, Výstupy lze spojit, nutný upínací odpor na Vcc, Montážní součin – Wired-AND; Třístavový výstup – Tri-state output (TS), Na výstupu hodnoty 0, 1, Z (vysoká impedance-odpojeno), Výstupy lze spojovat, Řízení výstupních členů zajistí, že pouze jeden vysílač není v Z

70.Vytváření společné sběrnice (spojování výstupů)
Standardní nelze spojovat, Otevřený a třístavový ano.

71.Záchytný registr s třístavovým výstupem (Latch, TS)

D Latch with TS output

$\overline{OE}$	LE	D^i	Q^{i+1}
0	1	0	0
0	1	1	1
0	1	X	Q^i
1	X	X	Z

72.Obousměrný budič sběrnice (Transceiver)

Bus Transceiver

$\overline{OE}$	DIR	A port	B port
0	0	A → B	Z
0	1	Z	B → A
1	X	Z	Z

73.Společná sběrnice – použití, možnosti vytvoření (OC, TS)
Použití: propojení komunikujících bloků počítače, Sběrnice jednosměrná nebo obousměrná, Sběrnice s třístavovými budiči nebo s budiči s otevřeným kolektorem.

74.Hazardy v logických obvodech, co je hazard, jak vzniká
Hazard je krátká neočekávaná změna výstupního signálu (glitch), která není matematickým výstupem logické funkce. (Statický hazard – výstup logického obvodu má být trvale v 0 nebo 1 (má být statický), místo toho se objeví krátký impuls do opačné úrovně.; 0-1-0 … statický hazard v 0; 1-0-1 … statický hazard v 1)

75.Zjištění hazardu

Signál ze vstupu logického obvodu se šíří na výstup různými cestami, které se někdy rozdělí a pak zase spojí. Signál se různými cestami vlivem časového zpoždění na hradlech a vodičích šíří různou dobu. V místě opětovného spojení má signál z různých cest různý časový posun.

76.Kdy je hazard kritický

Hazardy v kombinačních obvodech nejsou kritické – výstup kombinačního obvodu se po určité (krátké) době vždy ustálí ve správné hodnotě, Hazardy v sekvenčních obvodech mohou uvést klopné obvody do nesprávného stavu a tím nastavit celý sekvenční obvod (konečný automat) do nevratného kritického stavu!!!

77.Jak hazardům předcházet

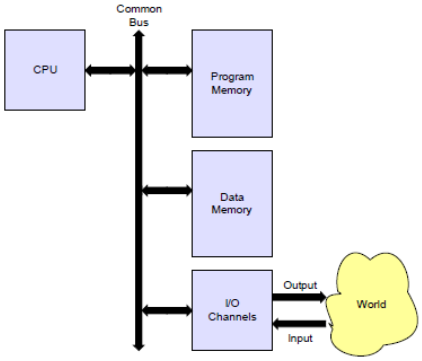
Synchronní návrh a správný výpočet maximální povolené hodinové (synchronizační) frekvence.

78.Systémová struktura počítače

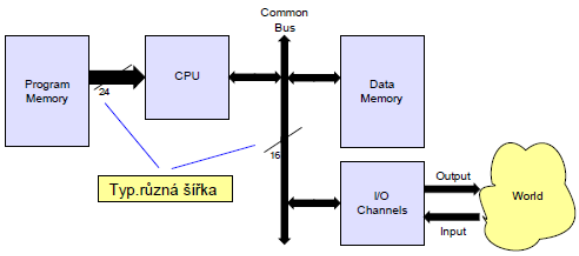
Počítač – sériový stroj: CPU – Procesor, Paměť programu, Paměť dat, Vstup / výstup, Společná sběrnice, Hodiny (synchronizace), XTAL – krystal, Nulování, Periferie, Vnější paměť, Master sběrnice

79.Architektura „von Neumann“ a „Harvard“ – základní rozdíly a vlastnosti

„von Neumann“: Jednodušší struktura, Sdílená sběrnice – nelze paralelně transportovat instrukce a data, Společný paměťový prostor instrukcí a dat. Možno pružně rozdělit paměť pro instrukce data a zásobník (pokud vše v RAM), Prostor pro zásobník (Stack) bývá dostatečný.



„Harvard“: Složitější struktura, Oddělené sběrnice – možnost paralelního transportu instrukce a dat (rychlost větší), Oddělené prostory instrukcí a dat. Paměť programu (instrukcí) často širší slovo proti paměti dat – kompaktnější jednocyklové instrukční kódy., Zásobník často mimo paměť dat (rozdílná šířka paměti programu a dat). Hloubka zásobníku omezená (pak nutno přemístit do paměti dat – a to pomalé)



80.Společná sběrnice a její členění

Neumožňuje paralelní přístup, členění: Control bus, data bus, Address bus

81.Hodiny procesoru – účel

Vysílají hodinový (synchronizační) signál, který určuje „rychlost“, aby nedošlo k hazardu.

82.Signál RESET – účel a následky v počítači
Nastaví paměť na základní adresu – nejčastěji na 00

83.Power monitor – účel
Monitorovat, jestli je správné napětí?

84.Uložení instrukcí programu a proměnných programu
Oboje uloženo v paměti programu, instrukce se postupně dostávají do čítače instrukcí.

85.Základní části programového vybavení počítače
Nevím, co je myšleno otázkou, ale řekl bych, že: Firmware, Operační systém, Vývojářský software, Aplikace, a nebo by se teoreticky dalo vypsat: ALU – aritmetická jednotka, PC – čítač instrukcí, SP – ukazatel zásobníku, PSW – stavové slovo procesoru, Střadač, Adresový prostor univ. Registrů, Adr. prostor paměti programu, Adresový prostor paměti dat, Adresový prostor zásobníku, Adr. prostor vstupů/výstupů, Umístění tabulky vekt.přerušení

86.Instrukční cyklus – jeho části
Čtení instrukce, Vykonání instrukce, Dekódování instrukce, Čtení operandu, Zpracování operandů, Zápis výsledku, Řídicí sběrnice (CB), Datová sběrnice (DB), Adresová sběrnice (AB), Sběrnicový cyklus, Směr přenosu informace

87.Provedení instrukce – členění na části
Dekódování instrukce, Čtení operandu, Zpracování operandů, Zápis výsledku

88.Sběrnicový cyklus

89.Komunikace mezi bloky počítače
CB, DB, AB; Obousměrná sběrnice; Jednosměrná sběrnice; Pouze jeden bus master; Směr „čtení“; Směr „zápis“; Adresa platná; Data platná; Dekodér adresy; Pole paměťových jednotek; „čtení/zápis“ z/do paměti

90.Rozdělení společné sběrnice (AB,DB,CB) – účel
Zajímavá otázka: rychlejší přístup?

91.Určení směru přenosu po sběrnici
Určuje signál nWrite ?

92.Určení platnosti adresy a dat
Další instrukce se čte vždy z adresy uložené v PC a mění se po přečtení instrukce, lze změnit vykonáním podmíněného skoku

93.Dekodér adresy paměti a vstupů/výstupů - účel

94.Řízení toku programu – prostředky
loop, bra, call ?

95.Čítač instrukcí, registr instrukce – účel a činnost
Jsou v něm uloženy instrukce(jejich adresy) v pořadí, v jakém se budou vykonávat. Další instrukce se čte vždy z adresy uložené v PC a mění se po přečtení instrukce, lze změnit vykonáním podmíněného skoku, Adresa se dá změnit vykonáním skoku.

96. Realizace větvení programu
Prostě se to rozepisuje na menší a menší části.

97. Dekompozice řešeného problému – způsob a prostředky (procedura)
Viz 96

98. Volání procedury (synchronní) – princip, prostředky
Podívám se do čítače instrukcí, přečtu adresu, zapíšu na zásobník návratovou adresu, provedu instrukci, podívám se na ukazatel zásobníku, přečtu adresu, a jdu na ní.

99. Zásobník návratových adres, ukazatel zásobníku, vrchol zásobníku
Uchovává si návratové adresy, ukazatel mi vždy ukazuje na poslední přidanou – tedy na vrchol

100. Návratová adresa a return
Návratová adresa mi říká, kam se mám vrátit po skončení procedury(instrukce). Return musí být na konci každého CALL, aby se mi program vrátil na původní místo, odkud jsem ho zavolal (resp. o adresu +1)

101. Řízení toku programu – sekvenční, větvení, skok a volání (rozdíl)

102. Vnořené volání procedury a hloubka zásobníku

103. Sdílené prostředky procesoru, kontext
Sdílené prostředky jsou: Čítač instrukcí a ukazatel zádobníku

104. Střadač, stavové slovo procesoru

105. Souhrn akcí při synchronním volání procedury
Volání procedury je vyvolané programem (synchronní) ne vnější událostí (vnější událost – viz. přerušení), Stejným mechanismem se řídí i vnořené volání (procedura volá proceduru), Další instrukce se vždy čte z adresy právě uložené v čítači instrukcí (PC), Čti instrukci “Call“, Ulož “návratovou adresu“ (tj. obsah čítače instrukcí) do zásobníku, Vlož do čítače instrukcí počáteční adresu procedury, Ulož kontext do zásobníku, Proveď tělo procedury, Vyzvedni kontext, Proveď instrukci “Return“, Ta vyzvedne “Návratovou adresu“ ze zásobníku do čítače instrukcí (PC), Pokračuj v programu za místem volání “Call“ na pozadí – tj. čti instrukci z adresy uložené v PC

106. Volání procedury (asynchronní) – princip, prostředky
Princip: Dokonči právě prováděnou instrukci (instrukce je nepřerušitelná), Ulož (Push) “návratovou adresu“ do zásobníku (tj. adresu, která je právě v čítači instrukcí (PC)), Vyzvedni adresu ISR (tj. podprogramu obsluhy přerušení) z tabulky vektorů přerušení, Spusť ISR, Vynuluj “Interrupt Request Flag“ (závisí na typu procesoru), Ulož kontext do zásobníku, Proveď tělo ISR (vlastní obsluhu žádosti o přerušení), Obnov původní kontext (vyzvedni ho ze zásobníku), Proveď instrukci “Return“, Ta vyzvedne “Návratovou adresu“ ze zásobníku do čítače instrukcí (PC), Pokračuj v programu na pozadí – tj. čti instrukci z adresy uložené v PC
Prostředky: Systém přerušení, Hardwarové volání procedury, Předdefinovaná cílová adresa, Vektor přerušení, Tabulka vektorů přerušení, Asynchronní žádost o přerušení, Řadič přerušení, Vstupy žádostí o přerušení, Asynchronní událost, Žádost o přerušení do CPU, Potvrzení žádosti od CPU

107. Systém přerušení a řadič přerušení (princip)
Řadič přerušení na vstupech odchytává žádosti o přerušení a za výstupem jsou dva spínače – první: zda je přerušení povoleno a pak druhý- povolení maskovatelných žádostí.

108. Hardwarové volání procedury, předdefinovaná cílová adresa
109. Tabulka vektorů přerušení
110. Asynchronní událost a žádost o přerušení

111. Reakční doba přerušení, nepřerušitelná instrukce
Za jak dlouho se to přerušení nejdéle provede, nepřerušitelné instrukce jsou takové, které nelze přerušit, protože by došlo k nenapravitelné chybě

112. Program řízený událostmi
Viz 106 – princip

113. Priorita přerušení
High, low

114. Typy přerušení a vlastnosti (nemaskovatelné, maskovatelní, ladící-trap)
Jednohladinové=neprioritní, Vícehladinové:Nemaskovatelné, Softwareové, Maskovatelné,S pevnou HW prioritou, S dynamicky volitelnou prioritou

115. Obsluha přerušení – ISR
Interrupt service – když ho chceme používat, tak musí být povolen, nevolá se synchronně

116. Návratová adresa z přerušení, instrukce návratu z přerušení
Adresa, na kterou skočím zavoláním return na konci procedury, uložená na vrcholu zásobníku

117. Pozadí programu (background)
Nevím, ale myslím, že kontext a pozadí bude to stejný.

118. Kontext programu a přerušení (vztah)
Vnitřní stav PC, který se uloží do zásobníku, dojde-li k vyvolání obsluhy přerušení

119. Volání procedury (asynchronní) – princip a prostředky
Viz 120

120. Obsluha žádosti o přerušení – souhrn akcí
Žádost o přerušení, dokončení aktivní instrukce, uložení návratové adresy do zásobníku, vyzvednutí adresy obslužného programu ISR, Spuštění obslužné procedury přerušení, uložení kontextu do zásobníku, provedení těla obslužné procedury, obnovení kontextu, vyzvednutí návratové adresy, obnovení běhu přerušeného programu

121. Princip programu řízeného událostmi (Event Driven Program)
Viz 106 – princip

122. Programátorský model počítače
ALU – aritmetická jednotka, PC – čítač instrukcí, SP – ukazatel zásobníku, PSW – stavové slovo procesoru, Střadač, Adresový prostor univ. registrů, Adr. prostor paměti programu, Adresový prostor paměti dat, Adresový prostor zásobníku, Adr. prostor vstupů/výstupů, Umístění tabulky vekt.přerušení

123. Typy adresových prostorů

Accumulator, Register, Program memory, program counter, stack pointer, procesor status, data memory, Input/Output space, HW stack(CPU)

124. Mapování vstupů a výstupů do paměti (Memory Mapped I/O) – princip

V program memory je vyhrazen prostor pro High priority a Low priority přerušení, v data memory je I/O paměť

125. Universální registry a registry speciálních funkcí (SFR) – účel

GPR - univerzální, SFR – slouží jako ovládací registry periferií

126. Umístění zásobníku – pojem hardwarový zásobník

Slouží k uložení informací o běhu programu, má metody PUSH a POP(PULL), nevýhoda zásobníku s pevnou délkou: může přetéct

127. Architektura CPU a operandy

Orientovaný na zásobník: ALU, PSW (Příznakový reg), zásobník, Střadačově orientovaný: ALU, PSW (Příznakový reg), Paměť dat; Registrově orientovaný: ALU, PSW (Příznakový reg), univerzální registry (v CPU); Paměťově orientovaný: ALU, PSW, paměť dat

128. Příznakový registru (PSW) – účel

Slouží k nastavení chování registru: povolení přerušení, směr čtení, Příznaky se nastavují, pokud operace dopadne dobře, …

129. Způsoby adresování operandů (přímá adr. nepřímá adr., indexové adr.,…)

Přímá adresace: operand je přímo částí instrukčního slova, Nepřímé adresování lze použít v celém rozsahu datového paměťového prostoru. Přes pomocný registr se dohledá správná adresa, na kterou se má jít. Indexová adresace se váže na adresu v paměti např. jede třeba od 80 do 85 všechny místa v paměti.

130. Adresování pomocí bank – princip a účel

Používá se, když je větší paměť, tak se rozdělí na několik částí – bank za účelem rychlejšího přístupu a taky proto, že do velikosti jedné banky by se nám složitější program nemusel vejít. V každé instrukci se pak musí napsat, v jaké bance chceme „operovat“

131. Přemapování paměti – princip a účel

Slouží k vypnutí bank v paměti a používá se pouze banka 0. Není tedy nutné psát do příkazů, v jaké bance pracujeme, protože je povolena pouze jedna

132. Instrukční soubor – základní dělení

Pevná řádová čárka, Přesuny operandů, Aritmetické operace, Posuny a rotace, Logické operace, Bitové operace, Nepodmíněné skoky, Absolutní, relativní, Podmíněné skoky, Volání podprogramu a návrat, Řídící instrukce

133. Instrukční soubor – operace aritmetické, logické, posuny, skoky a volání, bitové,.

Aritmetické operace: ADDxy – sečti, ADDxyC - sečti včetně carry, SUBxy – odečti, SUBxyB - odečti včetně borrow, MULxy – násob, INCx – inkrementuj, DECx – dekrementuj, NEG - x = -x (dvojkový doplněk), DA - dekadická korekce, CP - porovnej (komparuj)(odčítání bez uložení výsledku), TST – testuj; Logické operace: ANDxy – AND, IORxy - OR (inclusive or, nevýhradní nebo), XORxy - XOR (exclusive or, výhradní nebo), COM - x = NOT x; Rotace: RLNC - rotuj vlevo bez carry (C), RLC - rotuj vlevo přes carry (C), RRNC - rotuj vpravo bez carry (C), RRC - rotuj vpravo přes carry (C); Posuny: Posun vlevo (logický je shodný s aritmetickým) C=0 pak RLC, Posun vpravo – logický C=0 pak RRC,

Posun vpravo – aritmetický Nejvyšší bit do C pak RRNC; Bitové operace: BCx - nuluj bit, BSx - nastav bit, BTxSC - testuj bit (skip if clear), BTxSS - testuj bit (skip if set), BTG - otoč bit (bit toggle); Nepodmíněné skoky: GOTO - přejdi na (absolutní adresa), celý prostor pam.prog., BRA - přejdi na (relativní adresa -1024, +1023), Podmíněné skoky: Testují bity stavového slova (C, N, Z, OV) BC - Branch if Carry (C=1), BNC - Branch if Not Carry (C=0), BN - Branch if Negative (N=1), BNN - Branch if Not Negative (N=0), BZ - Branch if Zero (Z=1) (výsledek je 0), BNZ - Branch if Not Zero (Z=0) (výsledek není 0), BOV - Branch if Overflow (OV=1), BNOV - Branch if Not Overflow (OV=0) Volání podprogramu: CALL - absolutní adresa, celý prostor paměti programu, RCALL - relativní adresa; Návrat z podprogramu: RETLW - návrat z procedur, hodnota ve WREG, RETURN - návrat z podprogramu, RETFIE - návrat z ISR, povolí přerušení od dané hladiny; Přesuny operandů: MOVxy – přesuň, LFSR – nastav, SWAP - zaměň půlslabiky (nibble), CLR – nuluj, SET - nastav na FFh, POP - vyzvedni položku z vrcholu zásobníku, PUSH - ulož položku na vrchol zásobníku; Přesuny operandů: TBLRD - table read (z FLASH), TBLWR - table write (do FLASH)

134. Podmíněné skoky – účel a realizace
Testují bity stavového slova (C, N, Z, OV)

135. Assembler – účel, základní části a základní vlastnosti
Jazyk symbolických instrukcí (též Jazyk symbolických adres), Nejbližší vyšší stupeň nad strojovým kódem, Používá symbolická jména instrukcí, proměnných, konstant, adres paměti programu i dat, Umožňuje využít všechny vlastnosti daného typu počítače a jeho instrukčního souboru, Nekontroluje téměř žádné nedovolené (nevhodné) kroky programátora narozdíl od vyšších programovacích jazyků – JAVA, C), Používá se v časové kritických částech programů, nebo tam kde danou operaci nelze naprogramovat ve vyšším jazyku, V řídicích aplikací je typická kombinace částí programu v assembleru a částí v jazyku C, Program v assembleru je nepřenosný na jinou platformu, je vázaný na instrukční soubor určitého typu procesoru

136. Direktivy assembleru – účel
Řídit překlad, Označit začátek programu, Označit konec programu, Určit umístění programu v paměti programu, Určit umístění proměnných v paměti dat, Založit symbolická jména zvolených míst v programu, Založit symbolická jména proměnných, Zajistit inicializaci (nast. počáteční hodnoty) vybraných proměnných, Založit symbolická jména konstant a nastavit jejich hodnoty, Vkládat do programu konstanty v různých formátech (dec,hex,bin), Nastavit konfigurační bity počítače, Vkládat (zatahovat) do programu další části programu ze souboru, Označit části programu (moduly) symbolickým jménem (Makro) a pod tímto jménem moduly v programu používat.

137. Polyadické číselné soustavy – princip a definice
Poziční číselné soustavy, které jsou definovány tzv. základem, jenž určuje maximální počet číslic vyskytujících se v dané soustavě a existují zde pouze nezáporná čísla. Ex.: dvojková, desítková, šestnáctková, …

138. Základ číselné soustavy, číslice a váha pozice v čísle
Základ je počet čísel vyskytujících se v soustavě =z, váhy – desítková: každé číslo je $10^{\text{na číslo}}$, dvojková: 0-1,1-2,2-4,3-8,…

139. Číselné soustavy používané v počítačové technice a jejich vlastnosti
Převážně dvojková a šestnáctková. A=10,F=15

140. Převod mezi dvojkovou a desítkovou soustavou
Dvojkové číslo si zapíšu jako dvě na něco, kde první místo je 2^0 a doleva vždy +1 a doprava -1

141. Převod mezi desítkovou a dvojkovou soustavou (celé a zlomkové části)

Dělíme postupně základem soustavy (z = 2) a zbytky píšeme jako koeficienty a_0, a_1 – tedy odzadu Zlomková část: násobíme postupně základem (z=2) a celé části jsou postupně rovny a_0, a_1– tedy píšeme je normálně(od nuly). Pozor: Číslo zapsané v desítkové soustavě konečným počtem číslic může vést v jiné soustavě na nekonečný počet číslic nebo na počet číslic pro které nemáme dostatek pozic (šířka slova) - ztráta přesnosti.

142. Převod mezi dvojkovou a šestnáctkovou soustavou a zpět

Jedna šestnáctková číslice odpovídá čtyřem číslicím dvojkovým($(z_2)^4=z_{16}$> převod je naprosto triviální. 2 -> 16: Rozdělím si číslice na skupiny po 4 a jen přepisuju. 16 -> 2: Každou šestnáctkovou číslici přepisuju na ekvivalent dvojkové

143. Řádová mřížka, modul, jednotky ř.m., délka ř.m.

Definuje formát čísel zobrazitelných v počítači-tj. definuje nejvyšší a nejnižší řád. Modulo: nejmenší číslo, které již v ř.m. není zobrazitelné, Jednotka: nejmenší zobrazitelné číslo, Délka: počet řádů obsažených v ř.m.

144. Zobrazení záporných čísel – přímý, aditivní a doplňkový kód – vlastnosti

Přímý kód: Aditivní kód: Dvojkový doplněk:

S(x)	x
0 0 0	+0
0 0 1	+1
0 1 0	+2
0 1 1	+3
1 0 0	-0
1 0 1	-1
1 1 0	-2
1 1 1	-3

x	B(x)
+3	1 1 1
+2	1 1 0
+1	1 0 1
0	1 0 0
-1	0 1 1
-2	0 1 0
-3	0 0 1
-4	0 0 0

x	C(x)
+3	0 1 1
+2	0 1 0
+1	0 0 1
0	0 0 0
-1	1 1 1
-2	1 1 0
-3	1 0 1
-4	1 0 0

145. Výpočet doplňkového kódu (Two's Complement Code)

Slouží k vypočítání záporné hodnoty daného čísla. Vypočítá se znegováním kladného čísla a přičtením jedničky.

146. Sčítání čísel bez znaménka

Napíšu si čísla pod sebe a normálně dle pravidel booleovské algebry sčítám. Jestliže mi to na nejvyšším řádu přeteče, tak je C=1.

147. Odčítání čísel bez znaménka

Prostě odečítám a pokud odečtu od nuly jedničku, tak je přenos.

148. Příznakové bity přenosu, výpůjčky, přeplnění a nulového výsledku – účel a použití

Číslo, které vyjde mimo formát dostane příznak OV (Overlow), Carry – přenos do vyššího řádu,

149. Sčítání a odčítání čísel se znaménkem

Když je záporný, tak neguju a přičítám jedničku.

150. Sčítání a odčítání čísel s šířkou větší než je šířka ALU

Pokud přeteče, tak se nastaví OV na jedničku.

151. Násobení čísel bez znaménka

Druhým číslem odzadu násobím první a součiny sepisuju pod sebe vždy s uskočením o jedno místo doleva. Na konci vše sečtu.

152. Násobení čísel se znaménkem

Viz 151, akorát v posledním kroku zneguju a přičtu jedničku .

153. Dělení čísel bez znaménka

Normálně dělím, jako v desítkové soustavě

154. Chyby vzniklé operacemi v řádové mřížce s omezenou délkou, zaokrouhlování

Při přeplnění formátu směrem nahoru (přes n) vznikají neodstranitelné (hrubé) chyby. Pro správnou funkci je nutné řádovou mřížku rozšířit (zvětšit n). Při přeplnění formátu směrem dolů (pod –m) dochází ke ztrátě přesnosti. Velikost chyby můžeme ovlivnit: Rozšířením ř.m. směrem dolů (zvětšit m), Způsobem zaokrouhlování výsledku.

155. Pohyblivá řádová čárka (Floating Point) – princip a vlastnosti

Čísla v pohyblivé řádové čárce (Floating Point) mají tvar: $x_z=m*z^e$, kde: m – mantisa (mantissa, significand), e – exponent (characteristic, exponent), z – základ číselné soustavy;
m – mantisa obsahuje informace o „hodnotě“ čísla, e - exponent obsahuje informace o pozici řádové čárky, k vyjádření **m** a **e** používáme kódy pro zobrazení záporných čísel. Čísla v pevné řádové čárce mají značně omezený rozsah hodnot. Pro zvětšení rozsahu hodnot používáme formát pohyblivé řádové čárky. Standard ANSI a dva IEEE(single-127, double-1023).

156. Normalizace čísel v pohyblivé řádové čárce – princip a účel

V podstatě to převedu na desetinný číslo, potom to převedu na 2 na něco, to něco převedu do binárky a nakonec přičtu 127 nebo 1023, dle fomátu. Ta normalizace je převedení exponentu u dvojky na binárku.

157. Skrytá jednička v pohyblivé řádové čárce – princip a účel

normalizovaný tvar nenulového čísla má v nejvyšším řádu |M| vždy jedničku — tu lze ze zápisu vypustit ⇒ lze použít princip skryté jedničky

158. Sčítání a odčítání v pohyblivé řádové čárce

Provádí se tak, že se srovnají exponenty a přičtou, či odečtou mantisy.

159. Násobení v pohyblivé řádové čárce

Sečtou se exponenty a vynásobí mantisy.

160. Dělení v pohyblivé řádové čárce

Odečtou se exponenty a vydělí mantisy.

161. Porovnání čísel v pohyblivé řádové čárce

Srovnají se exponenty a porovnají mantisy.

162. Paměť počítače – způsob přístupu (libovolný, sekvenční, LIFO, FIFO, CAM, Cache)

Libovolný(náhodný): položka se vybírá dle adresy; Sekvenční (postupný): Výběr položek postupně – sériová paměť; Zásobník (LIFO): Last In First Out; Fronta (FIFO): Firt In First Out;CAM(Contents Adressable Memory): VELMI rychlá asociativní paměť; Cache:

163. Paměť počítače – SRAM, DRAM, SDRAM, EPROM, EEPROM, FLASH – vlastnosti

Volatilní: po ztrátě napětí ztrácejí informaci, jsou rychlé na zápis i čtení, SRAM – Statická RAM, DRAM – Dynamická, SDRAM – Synchronní dynamická; Nevolatilní: po vypnutí informace zůstává, rychlé pro čtení, ROM, EPROM-Erasable, EEPROM-Electronically Erasable), FLASH

164. Paměť počítače – virtuální paměť – princip, logické a fyzické adresy

Jedná se o paměť, jejíž prostor je rozšířen na disk => logické adresy mohou mít hodnotu větší, než odpovídá operační paměti, za běhu je část instrukcí,které nejsou potřeba k provádění programu, odložena na disk (Swap), Paměťová jednotka, která přenáší mezi virtuální a fyzickou pamětí se nazývá stránka (Page) – toto je tzv. „stránkování paměti“

165. DMA přenos – princip, výhody proti přerušení.

Tento přenos je charakteristický tím, že dvě zařízení spolu pracují „bez“ využití procesoru. Aby se zbytečně neplýtval procesor na jednoduchou činnost kopírování dat z jednoho na druhé, tak pouze zinicializuje přenos a dostane zprávu, jakmile je proces dokončen. Může mezitím vykonávat jiné operace a není zbytečně vytížen. Přenos dat je navíc rychlejší díky tomu, že není nutné zapojovat do činnosti „pomalý“ procesor.

166. Procesory CISC a RISC – charakteristika..

CISC – Complex Instruction Set Computer: počítač s rozsáhlým souborem instrukcí (nebo též počítač s úplným instrukčním souborem), Mnoho instrukcí, jednoduchých (základních) i složitých, Přesun složitých operací ze software do hardware, Návrh CPU kompromisem ve směru - menší kód, větší Ci (CPI), Podpora vyšších programovacích jazyků (HLL – High Level Language) přenesena do hardware

RISC – Reduced Instruction Set Computer: Počítač s redukovaným souborem instrukcí, Menší počet základních instrukcí – jen nezbytné základní operace, Přesun složitých operací z hardware do software, Návrh CPU kompromisem ve směru – malé Ci (CPI), větší kód, Podpora vyšších programovacích jazyků (HLL – High Level Languge) přenesena do software (do kompilátoru)