Materiály na SZZ pro Informační Bezpečnost

Matěj Douša

Červen 2024

Obsah

1 Počítače a systémy

1.1 SP-19 (PA1)

Datové typy v programovacích jazycích. Staticky a dynamicky alokované proměnné, spojové seznamy. Modulární programování, procedury a funkce, vstupní a výstupní parametry. Překladač, linker, debugger.

Datové typy

V programovacích jazycích používáme proměnné, tedy něco, co uchovává datovou hodnotu s nějakou vnitřní strukturou. Proměnné jsou identifikovány svými jmény — identifikátory. Datový typ proměnné definuje vnitřní strukturu/reprezentaci dat a jejich význam. Tím určuje jakých hodnot může proměnná nabývat a také jaké operace lze s proměnnou (její hodnotou) vykonávat.

Jednoduché datové typy:

- celočíselné
  - existují různé délky — short, int, long (byte, long long, ...)
  - signed (znaménkové) — umí uložit i záporné hodnoty, používá se doplňkový kód
  - unsigned (neznaménkové) — ukládá jen kladné hodnoty, přímý kód
- s pohyblivou řádovou čárkou
  - existují různé délky — float, double, long double
  - znaménko (1 bit) + mantisa (velikost=přesnost) + exponent (velikost=rozsah)
- znakové

  Znaky jsou kódovány jako čísla, používá se ASCII / extended ASCII / UNICODE.
- logická hodnota

  Není v C, ale často se v jazycích vyskytuje (boolean — true/false).

Další datové typy:

- ukazatel (pointer)

  Adresy paměti, kde je uložen datový typ pointeru (pointer vždy ukazuje na konkrétní typ/funkci, případně void).
- výčtový typ (enum)
- struktura

  Je složena z dalších datových typů, klidně dalších struktur.
- union

  Ukládá více různých datových typů na stejné místo.
- třída (ve vyšších jazycích)

Statická a dynamická alokace

Staticky alokované proměnné:

- vzniknou běžnou deklarací
- ukládají se na zásobník (lokální proměnné) či do části .BSS (neinicializované globální proměnné) a .DATA (inicializované globální proměnné)
- v případě pole je nutno znát v době kompilace velikost (statická velikost)

Dynamicky alokované proměnné

- vzniknou použitím speciální funkcí/operátorem
- ukládají se na haldě (heap)
- přistupujeme přes pointer
- je možné alokovat paměť podle hodnot spočítaných za běhu programu

Spojové seznamy

- oproti poli nejsou položky seřazeny v paměti, ale každý prvek seznamu obsahuje ukazatel na další prvek.
- podobně jako v dynamicky alokovaném poli lze ukládat předem neznámý objem dat
- nelze jednoduše indexovat, ale lze libovolně přidávat či ubírat prvky z jakékoliv pozice v seznamu

Modulární programování

- složitější programy mohou být rozděleny do modulů
- tyto moduly lze použít v různých dalších částech programu
- modul má svou specifikační část (deklarace poskytovaných prostředků/rozhraní) a implementační část (definice/implementace poskytovaných prostředků)
- v C/C++ typicky hlavičkový soubor (.h/.hpp) a implementační soubor (.c/.cpp)

Procedury, funkce a parametry

- procedura/funkce je posloupnost příkazů uložených v paměti programu
  - procedura — bez návratové hodnoty (typ void)
  - funkce — s návratovou hodnotou
- použijeme ji zavoláním přes její jméno
- deklarace je specifikace jejího rozhraní — parametrů a typu návratové hodnoty
- definice je samotný kód funkce
- vstupní paramtery jsou informace, které využije kód funkce
- výstupní parametry jsou výsledkem běhu funkce — typicky se nějak změní a tím nám dají výsledek

Překladač

- překládá vyšší programovací jazyky do nižších
- ze zdrojového kódu vzniká objektový soubor — modul se strojovým kódem
- front-end přeloží konkrétní jazyk do vnitřní reprezentace (abstrakce nezávislá ani na platformě ani na jazyku)
- back-end přeloží vnitřní reprezentaci do strojového kódu konkrétní platformy

Linker

- spojuje přeložené moduly do výsledného celku — programu
- výstupem je spustitelný soubor

Debugger

- usnadňuje hledání chyb v kódu, také usnadňuje pochopení programu
- je vhodné kompilovat s informacemi pro ladění
- je možné si na nějakém místě běh programu zastavit a např. sledovat obsah proměnných, pouštět každý krok programu postupně...

1.2 SP-30 (SAP)

Kódy pro zobrazení čísel se znaménkem a realizace aritmetických operací (paralelní sčítačka/odčítačka, realizace aritmetických posuvů, dekodér, multiplexor, čítač). Reprezentace čísel v pohyblivé řádové čárce.

Čísla se znaménkem

Existuje několik možností, jak v počítači ukládat celá čísla:

- Přímý kód
  - první bit je znaménkový — určuje tedy, zda je hodnota za ním kladná či záporná
  - ostatní bity představují absolutní hodnotu čísla
  - existuj zde kladná i záporná nula
- Doplňkový kód
  - dle prvního bitu lze poznat znaménko čísla
  - převod kladné $\leftrightarrow$ záporné lze vysvětlit jako inverze bitů a následné přičtení jedničky
  - 0111 (7) $\rightarrow$ 1001 (-7)
  - není zde záporná nula
- Aditivní kód
  - uložené číslo je posunuto o nějakou konstantu, typicky polovina rozsahu
  - pro 4 bity určeme nulu jako 1000 — pak 1111 je 7, 0000 je -8
  - nula není zobrazena jako nula
  - není zde záporná nula

Čísla v pohyblivé řádové čárce

Reprezentace pohyblivé řádové čárky vychází ze zobrazení $A = M * z^e$ používaném např. ve fyzice, kde z je základ soustavy (zde 2), e je exponent jako celé číslo, M je mantisa.

- používá se normalizovaný tvar, tedy mantisa je zapsána tak, že ji nelze "posunout"více doleva.
- v přímém kódu mantisy je vlevo vždy jednička, která se skrývá (zvýšení přesnosti)
- pro mantisu se typicky používá přímý kód, pro exponent aditivní
- float (32b) typicky vypadá jako 1b znaménko, pak 8b exponent a nakonec 23b mantisa (tedy přesnost 24b)

Realizace aritmetických operací

- Paralelní sčítačka

  Tvořena více jednobitovými sčítačkami. Jednobitová sčítačka má 3 vstupy: A, B (sčítané bity) a vstupní přenos (carry — např. ze sčítačky nižšího řádu). Výstupy jsou S (výsledek) a výstupní přenos.
- Aritmetické posuvy

  Posun čísla vlevo/vpravo. Realizuje posuvný registr. Existuje více různých posuvů:
  - logický posuv — doplňuje nuly
  - cyklický posuv — doplňuje co vylezlo na druhé straně
  - aritmetický posuv — doplňuje 1 nebo 0 podle znaménka čísla
- Dekodér

  Kombinační logický obvod, který má méně bitů na vstupu než na výstupu, a podle tabulky převádí. Kodér má opačnou funkci.
- Multiplexor

  Na základě řídícího signálu vybere, který ze vstupů pošle na výstup (má několik vstupů + řídící vstup, a jeden výstup).
- Čítač

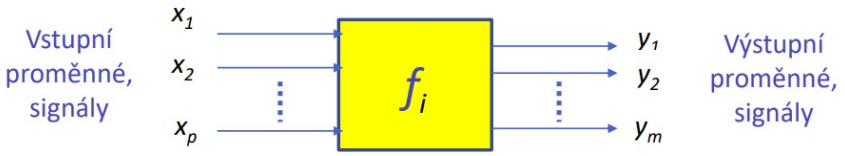
  Registr s funkcí inkrementu/dekrementu, může čítat nahoru a/nebo dolů. Existují úplné čítače (do mocnin 2) či neúplné (do jiných čísel). Typicky čítají v binárním kódu, lze i např. v Grayově kódu.

1.3 SP-28 (SAP)

Kombinační a sekvenční logické obvody (Mealy, Moore), popis a možnosti implementace na úrovni hradel. Minimalizace vyjádření logické funkce s využitím map.

Kombinační obvody

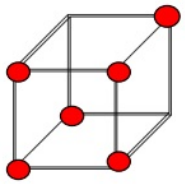
- popsány kombinační funkcí
- hodnoty všech výstupů (výstupních proměnných) jsou v každém časovém okamžiku určeny pouze vstupem (hodnotami vstupních proměnných) ve stejném okamžiku
- mohou být popsány např. Booleovskou (logickou) formulí
  Příklad: $f = x_1.\overline{x_2} + \overline{x_1}.x_2 = (x_1 + x_2).(\overline{x_1} + \overline{x_2})$
- obecně kombinační obvod vypadá následovně:



  Logická funkce $y_k = f(x_1, x_2, x_3, ..., x_p)$ existuje pro každý výstup y.
- možnosti reprezentace logických funkcí:
  - tabulka

ab	f
00	0
01	1
10	1
11	0

  - n-rozměrná krychle



  - Booleovský výraz
    Viz výše
  - mapa (Karnaughova)

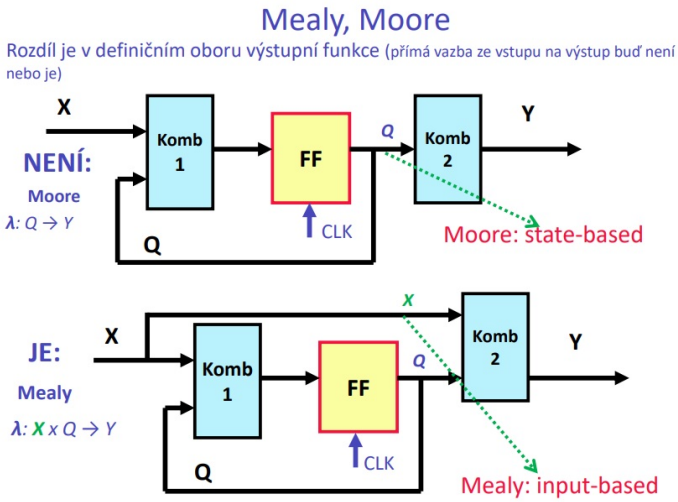
	a		b	
		X	1	
c		1	1	X

- možnosti realizace obvodů:
  - na úrovni hradel
  - mapování na technologii (FPGA, ASIC)
  - popis v jazyku (VHDL, Verilog)

Sekvenční obvody

- výstup závisí na posloupnosti/sekvenci hodnot na vstupu
- zapamatování se realizuje zpětnou vazbou
- popsány konečným stavovým automatem

- typy sekvenčních obvodů:
  - Moore
    Obvod, jehož výstup závisí pouze na vnitřním stavu.
  - Mealy
    Obvod, jehož výstup závisí také na aktuálním vstupu (kromě stavu).

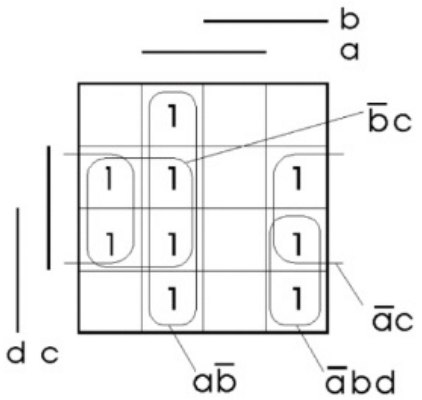


Implementace na úrovni hradel

- nejprve minimalizace logické funkce a zápis výsledku např. v Booleovském výrazu
- následně nakreslení/vytvoření funkce pomocí základních hradel — NOT, AND, OR, NAND, NOR, XOR

Minimalizace logické funkce

- smysl — zjednodušit a zkrátit zápis, snížit potřebný materiál pro výrobu
- minimalizace pomocí map — založenana hledání co největších skupin sousedních stavů.
- postup při minimalizaci:
  - vytvoření Karnaughovy mapy pro funkci
  - nalezení všech přímých implikantů (maximální skupiny jedniček či "dont care")
  - určení všech podstatných implikantů (obsahující jedničku, kterou jiný implikant neobsahuje)
  - pokud nejsou pokryty všechny vrcholy s "1", nutno vybrat další přímé implikanty (takové, kde je nejméně negací)



$a\bar{b} + \bar{b}c + \bar{a}c + \bar{a}bd$

- lze také kroužkovat nuly — pak je ale nutné funkci sestavit jinak.
  $(\bar{a} + \bar{b})(a + c + d)(a + b + c)$

1.4 SP-29 (SAP)

Architektura číslicového počítače, instrukční cyklus počítače, základní třídy souborů instrukcí (ISA). Paměťový subsystém počítače, paměťová hierarchie, skrytá paměť (cache).

Architektura číslicového počítače

- architektura se zabývá strukturou a chováním počítače
- řeší specifikaci různých funkčních modulů jako procesor a paměť a nebo např. instrukční sadu
- podsměry:
  - ISA — Instruction Set Architecture — architektura souboru instrukcí
  - Mikroarchitektura — konkrétní sestavení a složení procesoru
  - Systémový design — řeší další HW komponenty
- každý počítač je složen z následujících částí:
  - datová část procesoru — ALU, registry
  - řadič — řídící jednotka procesoru
  - paměťový subsystém
  - vstupní zařízení
  - výstupní zařízení
- typy architektury:
  - Von Neumannova architektura
    Data i instrukce jsou uložena spolu, nejsou explicitně označena/ny.
  - Harvardská architektura
    Data a instrukce jsou rozdělené.

Instrukční cyklus počítače

- čtení instrukce (IF — Instruction Fetch)
- dekódování instrukce (ID — Instruction Decode)
- načtení operandů (OF — Operand Fetch)
- provedení instrukce (IE — Instruction Execution)
- zapsání/uložení výsledku (WB — Write Back / Result Store)
- přerušení?

Co je instrukce? Obsahuje informace:

- co se má provést
- s čím se to má provést (operandy)
- kam se má uložit výsledek
- kde se má pokračovat

Tyto informace mohou být zadány explicitně, nebo mohou být dány typem instrukce, tedy architekturou počítače — tedy implicitně.

ISA — Architektura souboru instrukcí Co je potřeba určit:

- typy a formáty instrukcí, instrukční soubor
- datové typy, kódování a reprezentace, způsob uložení dat v paměti
- módy adresování paměti a přístup do paměti dat a instrukcí
- mimořádné stavy

Výhody:

- abstrakce — možnost různě implementovat stejnou architekturu instrukcí
- definice rozhraní mezi nízkoúrovňovým AW a HW
- standardizuje instrukce, bitové vzory strojového jazyka

Třídy souborů instrukcí (ISA)

- Střadačově (akumulátorově) orientovaná ISA

  Akumulátor je registr pro mezivýpočty, používá se implicitně jako zdroj pro výpočty i jako cíl pro výsledky. Používají se instrukce s jedním operandem. Nejstarší ISA (1949-60) — vyvinula se z kalkulaček.

  Výhody:

  - jednoduchý HW
  - minimální vnitřní stav procesoru — rychlé přepínání kontextu
  - krátké instrukce
  - jednoduché dekódování instrukcí

  Nevýhody:

  - častá komunikace s pamětí
  - omezený paralelismus mezi instrukcemi

  Populární v 50. — 70. letech, HW byl drahý, paměť byla rychlejší než CPU.

- Zásobníkově orientovaná ISA

  Pracovní registry jsou uspořádány do struktury zásobníku. Přistupuje se k vrcholu tohoto zásobníku. Využití pro vyhodnocení výrazů a vnořená volání podprogramů. Většina instrukcí nemá operand (použije se implicitně např. vrchní 2 registry zásobníku).

  Výhody:

  - jednoduchá a efektivní adresace operandů
  - krátké instrukce
  - krátké programy
  - jednoduché dekódování instrukcí
  - snadno lze napsat neoptimalizující překladač

  Nevýhody:

  - nelze náhodně přistupovat k lokálním datům
  - omezený paralelismus — zásobník je sekvenční
  - přístupy do paměti je těžké minimalizovat

- ISA orientovaná na registry pro všeobecné použití

  Dnes převládá. GPR — General Purpose Registers. Typicky 2 nebo 3 operandy.

  Výhody:

  - registry (a cache) jsou rychlejší než paměť
  - k registrům lze přistupovat náhodně
  - registry mohou obsahovat mezivýsledky a lokální proměnné
  - méně častý přístup do paměti

  Nevýhody:

  - složitější překladač (optimalizace pro použití registrů)
  - přepnutí kontextu trvá déle

Paměťový subsystém počítače

- cache (skrytá paměť) — rychlá, drahá, umístěna blíž k procesoru
- hlavní paměť — pomalejší, levnější, větší
- vnější paměť — pomalá, velká
- záložní paměť (CD, DVD, flash, magnetické pásky)
- RAM — random access memory (přístup adresou)
- CAM — content adressable memory (přístup klíčem)

Paměťová hierarchie

- registry
- L1 cache (SRAM)
- L2 cache (SRAM)
- L3 cache (SRAM)
- hlavní paměť (DRAM)
- HDD, SSD
- Mass storage (optical disks, tapes)
- Remote storage (cloud)

Cache

Kopie často používaných dat z hlavní paměti

- časová lokalita
  Data, ke kterým bylo právě přistupováno, budou pravděpodobně brzy potřeba znovu.
- prostorová lokalita
  Po přístupu k nějakým datům se pravděpodobně budou používat i vedlejší data.

1.5 OB-4 (APS)

2 Šifrování a sítě

3 Obecná bezpečnostní teorie

4 Matematika

5 Programování