Metody a algoritmy komprese dat

Od základních principů k aplikaci

P. Petyovský aka Poke,
poke@vkv.name

(rev.15)

Úvod do komprese dat

Definice

„Komprese dat (také komprimace dat) je speciální postup při ukládání nebo transportu dat. Úkolem komprese dat je zmenšit datový tok nebo zmenšit potřebu zdrojů při ukládání informací.“

„Komprimace dat je postup snažící se postihnout charakter dat a eliminovat jejich datovou redundanci.“

„Redundance obecně označuje takový stav nebo vlastnost, kdy je použito větší množství prvků než je obvyklé nebo nezbytně nutné.“

zdroj wikipedia

Pojem kompresní metoda a kompresní princip (kódování)

Existuje mnoho různých kompresních metod využívajících stejných kompresních (kódovacích) principů. Často je v literatuře uvedeno, že vznikla nová kompresní metoda (která, ale většinou využívá známé kompresní principy). Jen velmi zřídka, vznikne nový kompresní princip (využití markovovských modelu, metoda LZMA).

Kompresní principy se nezabývají implementačními detaily. Tuto část problému řeší až kompresní metoda, která je velmi často spojena s nějakým datovým formátem např.: `.zip, .arj, .rar,` apod...

Rozdělení kompresních principů

Základní dělení:

- **Ztrátové - transformace pomocí DCT (u JPEG), fraktální komprese, mezisnímková komprese (u MPEG)**
- **Neztrátové**
  - **Primární - RLE, LZW, LZ77**
  - **Sekundární - Huffmanova, Aritmetická komprese**

Dělení dle výpočetní náročnosti:

- **Symetrické**
- **Asymetrické**

Dělení dle počtu průchodů:

- **Jedno-průchodové**
- **Více-průchodové**

Kódování RLE

(bezztrátové, primární, jedno-průchodové, takřka symetrické)
Jednoduchá a pro velkou třídu dat i efektivní metoda, vychází z předpokladu, že ve vstupních datech, se opakují „sousední“ hodnoty. Tehdy do souboru zapíšeme informaci o počtu opakujících se hodnot a poté hodnotu samotnou.

Odtud také název metody: Run-Length Coding. Metoda pro kódování délky runu (skupiny dat majících stejnou hodnotu).

Příklad:
Vstupní data:

”AAAhoooooj”

Výstupní data:

”<3A>h<4o>j”

V praxi je navíc nutné odlišit běžná data od odkazů (počet, opakovaná hodnota), abychom při dekompresi byli schopni data správně dekódovat.

Kompresní metoda RLE

Aplikovaná v datovém formátu `.pcx` .
Při kódování pixelů v obraze definovaných jedním bytem rozlišíme příznak opakování hodnotou nejvyššího bitu:

1	*čítač*	*hodnota*	*hodnota* se opakuje *(1+čítač)* krát
0	*hodnota*		Výstup 7bitové neopakující se *hodnoty*
10000000	*hodnota*		Výstup pokud *hodnota* obsahuje MSB=1

Popis algoritmu:

1. Vynuluj *Čítač*
2. Do proměnné *BarvaA* přiřaď hodnotu pixelu.
3. Dokud jsou na vstupu pixely, opakuj:
   3.1 Do proměnné *BarvaB* přiřaď hodnotu dalšího pixelu
   3.2 Pokud *BarvaA == BarvaB* a zároveň hodnota *Čítače* nepřesáhla maximální hranici (danou velikostí bytu, slova), potom *Čítač = Čítač +1*
   Jinak:
   Zapiš *Čítač* a *BarvaA* na výstup
   Do proměnné *BarvaA* přiřaď hodnotu *BarvaB*
   Vynuluj *Čítač*
4. Zapiš *Čítač* a *BarvaA* na výstup

V případě, že kódovaný obrázek obsahuje neopakující se hodnoty v sousedních pixelech, dochází u metody RLE k *záporné kompresi*. Komprese RLE je vhodná pro obrázky kreslené ”od ruky” nebo tzv. ”Cartoons” – ilustrace s většími stejnobarevnými plochami.

Kódování LZ77

(neztrátové, primární, jedno-průchodové, asymetrické)

Původní myšlenka pochází z roku 1977 A. Lempel, J. Ziv. Metoda LZ77 bývá někdy v literatuře také označovaná jako LZSS (i když např. wikipedia označuje LZSS jako modifikaci LZ77).
Metoda se snaží nalézt ve vstupních datech opakující se sekvence symbolů a ty na výstup předat ve formě odkazu na předchozí (již odeslaná) data.
Odkazem je myšlen údaj o pozici opakující se sekvence v předchozích datech. Druhým důležitým parametrem odkazu je délka opakující se sekvence.

Příklad:

Vstupní data:

```
"the_rain_in_Spain_falls_mainly_in_the_plain"
".....ain.................................."
".....ain_................................."
```

Výstupní data: (V příkladu lze ušetřit ještě jeden byte, zjistěte kde?)

```
"the_rain_<3,3>Sp<9,4>falls_m<11,3>ly_<16,3><34,4>pl<15,3>"
```

V praxi je navíc nutné odlišit běžná data od odkazů, abychom při dekompresi byli schopni data správně dekódovat.
Možností je několik, běžně používanou je rozšíření symbolu o další bit, který rozhodne, zda jde o data nebo odkaz: `<pozice,délka>`. (Vhodné v případě následného využití některé z metod sekundární komprese.)
Metoda je jedno-průchodová což je výhodná vlastnost, zvláště při kompresi velkého objemu dat na hardwarové úrovni.
Metoda je silně asymetrická, výpočetní náročnost při kompresi je možné snížit použitím binárního vyhledávácího stromu.

Kódování LZW

(neztrátové, primární, jedno-průchodové, asymetrické)

Metoda LZ77 modifikována v roce1984 T. Welchem. Vzniklá metoda LZW bývá někdy označovaná jako *dictionary based encoding (slovníkově orientované kódování).* Metoda je patentována.

Princip vyhledání dříve se opakujících sekvencí zůstala zachována, modifikován je způsob kódování.

Metoda používá místo odkazů (pozice, délka) odkaz na položku slovníku, který je při kompresi vytvářen a který je neustále doplňován a jeho délka a počet položek tím narůstá.

Základní velikost slovníku je 512 položek a maximální velikost slovníku bývá nejčastěji 4096.

Po té je slovník smazán a metoda opět začíná se slovníkem velikosti 512 položek. Modifikací metody je systematické promazávání nejdéle nepoužitých slov/položek ve slovníku namísto smazání celého slovníku.

Metoda má v případě varianty s pravidelným mazáním celého slovníku, menší kompresní poměr než LZ77.

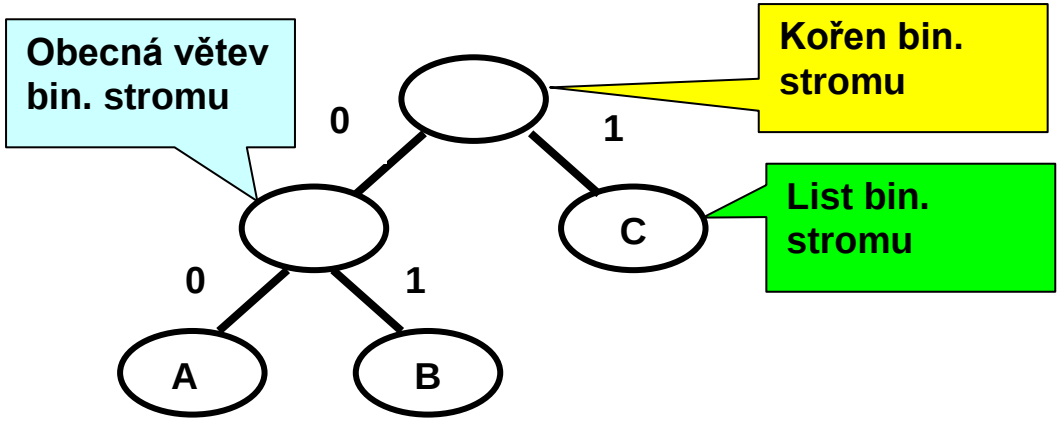
Huffmanovo kódování

(sekundární, neztrátové, mírně asymetrické, nejčastěji dvou-průchodové)

Myšlenka kódování pochází z roku 1952 od D. Huffmana. Je založena na použití různě dlouhých bitových kódů pro symboly s různou frekvencí výskytu. V praxi používána dávno před r. 1952 (např. Morseovka). (Nejčastěji používané vstupní symboly mají nejkratší výstupní kód.)
Principem je vytvoření binárního stromu na základě znalosti frekvencí výskytu jednotlivých symbolů ve vstupním souboru.
Příklad:
Vstupní data:

"CABACBBBCCCC"

Vstupní symbol	Počet výskytů
A	2
B	4
C	6



Vstupní symbol	Délka výstupního řetězce	Výstupní řetězec
A	2	00
B	2	01
C	1	1

Výstupní data (binárně): **"10001001010101111"**

Pro úspěšnou dekompresi je nutné znát tvar binárního stromu, proto je nutné:

- **Strom ke komprimovaným datům připojit**
- **Strom je fixně dán a je znám dopředu.**
- **Lze využít tzv. dynamickou tvorbu binárního stromu, není nutné jej přenášet, ale menší kompresní poměr.**

Otázka:
Jaký tvar bude mít binární strom v případě těchto frekvencí výskytu vstupních symbolů?

Vstupní symbol	Počet výskytů
A	2
B	2
C	3
D	3
E	4 (7)

Aritmetické kódování

(sekundární, neztrátové, mírně asymetrické, běžně dvou-průchodové, výpočetně náročné)

Myšlenka kódování je založena na úvaze, že bit není dál nedělitelné množství informace. Pracuje s četností výskytu vstupních symbolů jako s desetinnými čísly, proto velká výpočetní náročnost. Běžně se zde pracuje s desetinnými čísly majícími stovky a více bitů.

Zatímco u Huffmanova kódování měl vždy každý konkrétní vstupní symbol přiřazen svůj konkrétní počet bitů na výstupu. Aritmetické kódování pracuje vždy se skupinou několika vstupních znaků, které kóduje jako určitý počet výstupních bitů. Tím je zajištěn ještě vyšší kompresní poměr, než je tomu u Huffmanova kódování.

Zjednodušená představa je, že bychom v případě Huffmanova kódování (pomocí binárního stromu) definovali několik různých binárních stromů, které bychom průběžně střídali.

Metoda je bohužel patentovaná, proto je i přes výhody vysokého kompresního poměru prakticky nepoužívaná.

Reprezentace desetinného čísla v binární soustavě

Příklad:

$$(5.75)_{dec} = (101,11)_{bin}$$

$$(5*10^{0} + 7*10^{-1} + 5*10^{-2})_{dec} = (1*2^{2} + 0*2^{1} + 1*2^{0} + 1*2^{-1} + 1*2^{-2})_{bin}$$

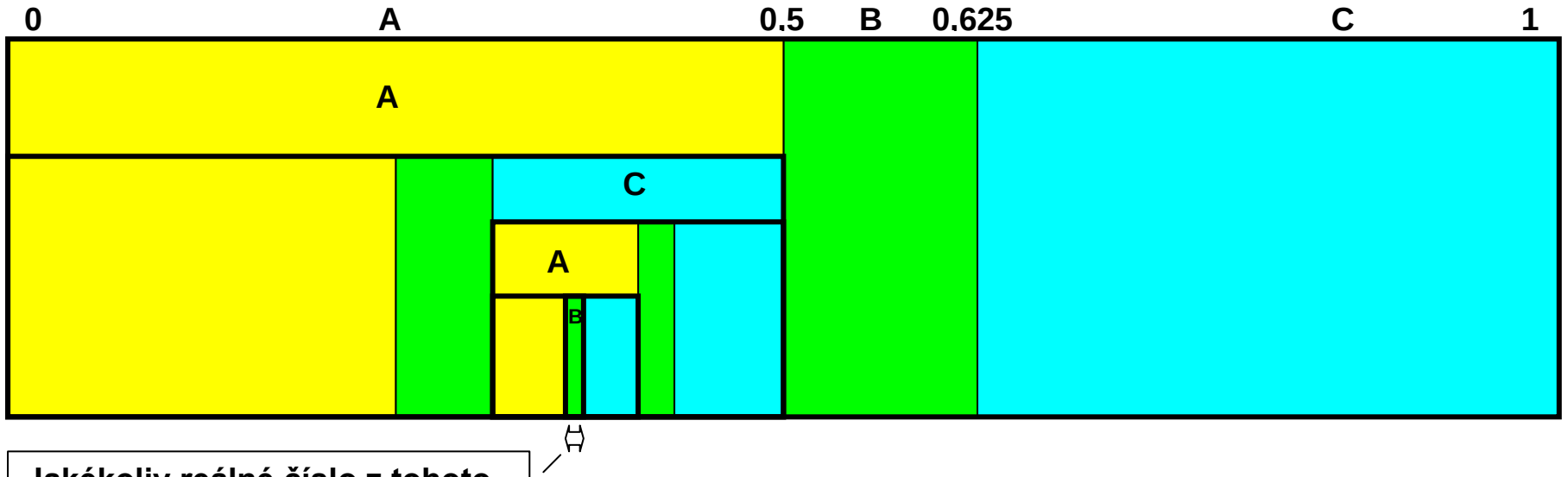
Pozn.:
Díky nesoudělnému základu desítkové a dvojkové soustavy platí, že desetinné číslo mající v jedné soustavě vyjádření s konečným počtem cifer, může být v druhé soustavě vyjádřené desetinným číslem s opakující se periodou.

Aritmetická komprese příklad

Vstupní data:

"ACABACAC"

Vstupní symbol	Počet výskytů	Četnost výskytů
A	4	4/8 = 0,5
B	1	1/8 = 0,125
C	3	3/8 = 0,375



Jakékoliv reálné číslo z tohoto intervalu kóduje sekvenci vstupních dat: ACAB

Aritmetická komprese numerická podoba příkladu

Počáteční hranice intervalů pro vstupní symboly (A,B,C) jsou:

$$\left\langle \frac{0}{8};\frac{4}{8};\frac{5}{8};\frac{8}{8}\right|.$$

Úprava hranic intervalů po symbolu A:

$$\left(\left\langle \frac{0}{8};\frac{4}{8};\frac{5}{8};\frac{8}{8}\right| - \frac{0}{8}\right)\cdot\frac{1}{2}+\frac{0}{8} = \left\langle \frac{0}{16};\frac{4}{16};\frac{5}{16};\frac{8}{16}\right|.$$

Úprava hranic intervalů po symbolu C:

$$\left(\left\langle \frac{0}{16};\frac{4}{16};\frac{5}{16};\frac{8}{16}\right| - \frac{0}{16}\right)\cdot\frac{3}{8}+\frac{5}{16} = \left\langle \frac{40}{128};\frac{52}{128};\frac{55}{128};\frac{64}{128}\right|.$$

Úprava hranic intervalů po symbolu A:

$$\left(\left\langle \frac{40}{128};\frac{52}{128};\frac{55}{128};\frac{64}{128}\right| - \frac{40}{128}\right)\cdot\frac{1}{2}+\frac{40}{128} = \left\langle \frac{80}{256};\frac{92}{256};\frac{95}{256};\frac{104}{256}\right|.$$

Poslední symbol je B, hledáme tedy takové číslo z prostředního intervalu, které dokážeme reprezentovat co nejlépe (tj. jako co nejkratším desetinným číslem v dané soustavě):

$$\left\langle \frac{92}{256};\frac{95}{256}\right| \rightarrow \frac{92}{256} = \frac{23}{64} = \frac{1}{4}+\frac{1}{16}+\frac{1}{32}+\frac{1}{64} = \frac{1}{2^2}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^6} = 0.010111_{bin}.$$

Výsledek je tedy „010111“. Pro úspěšnou dekompresi je stejně jako u Huffmanova kódování nutné znát četnosti výskytů vstupních symbolů.

CharPress

(Tompacker, TurboImploder, PkLite)

Autor byl člen brněnského Sinclair klubu s přezdívkou Char. Celý algoritmus naimplementoval jako modul do debuggeru DEVAST. Tento modul distribuoval MataSoft, který provedl několik modifikací.

TurboImploder využívá původní CharPress kód a vydává ho za svůj, podobně PkLite.
Tompacker představuje finální verzi kompresního nástroje vycházejícího z CharPress kódu (autoři: Char, Matasoft, Tom).

Využívá kódovacího principu LZ77. Implementace k rozlišení běžných dat od informace `<pozice,délka>` používá tzv. "bajtovou značku" (nejméně používaná hodnota ve vstupních datech). Implementace tedy vyžaduje dva průchody vstupními daty.

Velikost paměťového okna LZ77 metody: 11 bitů (2048), maximální délka kódována jako 5 bitů (31+3).
Výskyt značkovacího bajtu ve vstupních datech je na výstup kódován jako dvojbajtová hodnota:

`<bajtová značka,0>`.

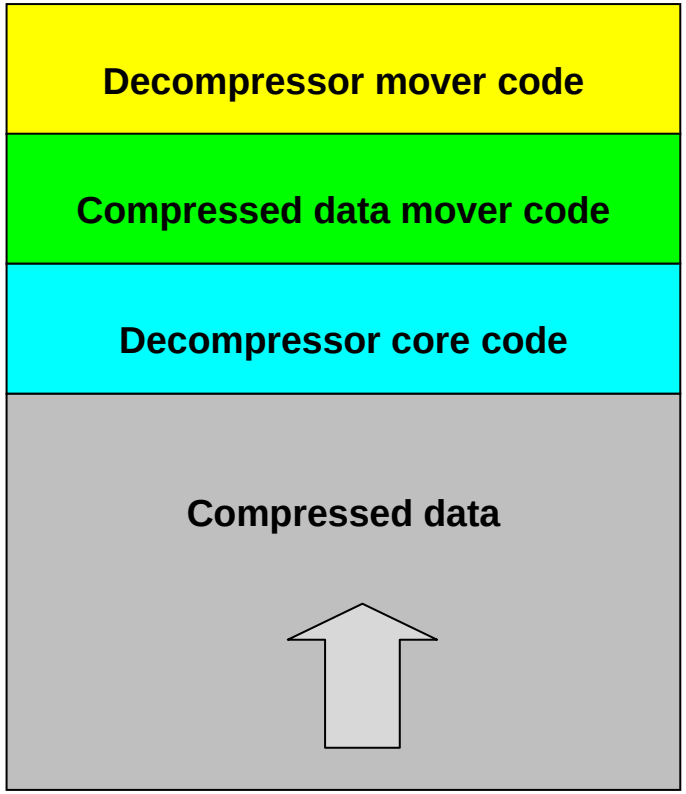
Implementace se musela vypořádat s nutností komprimace a dekomprimace dat v omezeném paměťovém prostoru. Průběžně dekomprimující se data přepisují již nepotřebné části dat komprimovaných.

CharPress metoda využívá modifikovaný způsob čtení dat. Data jsou komprimována v opačném pořadí, než jsou následně dekomprimována.

Implementace metody zachovává původní tvar nekomprimovaných dat, což umožňuje vícenásobnou komprimaci a tím vylepšení kompresního poměru.

CharPress dekompresor

```
          ld   hl,depacker_begin
          ld   bc,depacker_size
          ld   de,depacker_temporary_space
          pushde
          ldir
          ret
depacker_begin:
          PHASE    depacker_temporary_space
          ld   de,data_depacked_start
          ld   bc,data_packed_size
          ldir
          ex   de,hl
          ld   de,data_depacked_end
depacker_loop:
          dec  hl
          ld   a,(hl)
          dec  de
          ld   (de),a
          sub  data_packed_mark
          jr   nz,depacker_loop
          dec  hl
          or   (hl)
          jr   z,depacker_test
          dec  hl
          pushhl
          ld   l,(hl)                ;;; offset low byte
          ld   c,a
          and  07h
```



```
        ld  h,a                 ;;; offset high byte (only 3 bits)
        inc hl
        add hl,de
        xor c
        rrca
        rrca
        rrca
        add a,03h
        ld  c,a                 ;;; length of frame (5 bits + 3 values)
        lddr
        inc de
        pop hl
depacker_test:
        sbc hl,de
        add hl,de
        jr  nz,depacker_loop    ;;; is rest of data in the right place?
        IF  data_depacked_init = 0
         ret
        ELSEIF
         jp data_depacked_init
        ENDIF
depacker_end:
        DEPHASE
        DB  XXX
        DB  YYY
        DB  ZZZ
```

Utilita Cross CharPress

(Poke, dle CharPress)

Důvody vzniku:

- **Mnoho vývojářů dnes vyvíjí na PC**
- **CharPress je stále v ČR a SR oblíben pro svou rychlost dekomprese i efektivitu komprese**
- **Přímé využití výkonu dnešních procesorů namísto emulace původního kompresního Z80 kódu**
- **Automatizace zpracování (např. makefile apod.)**

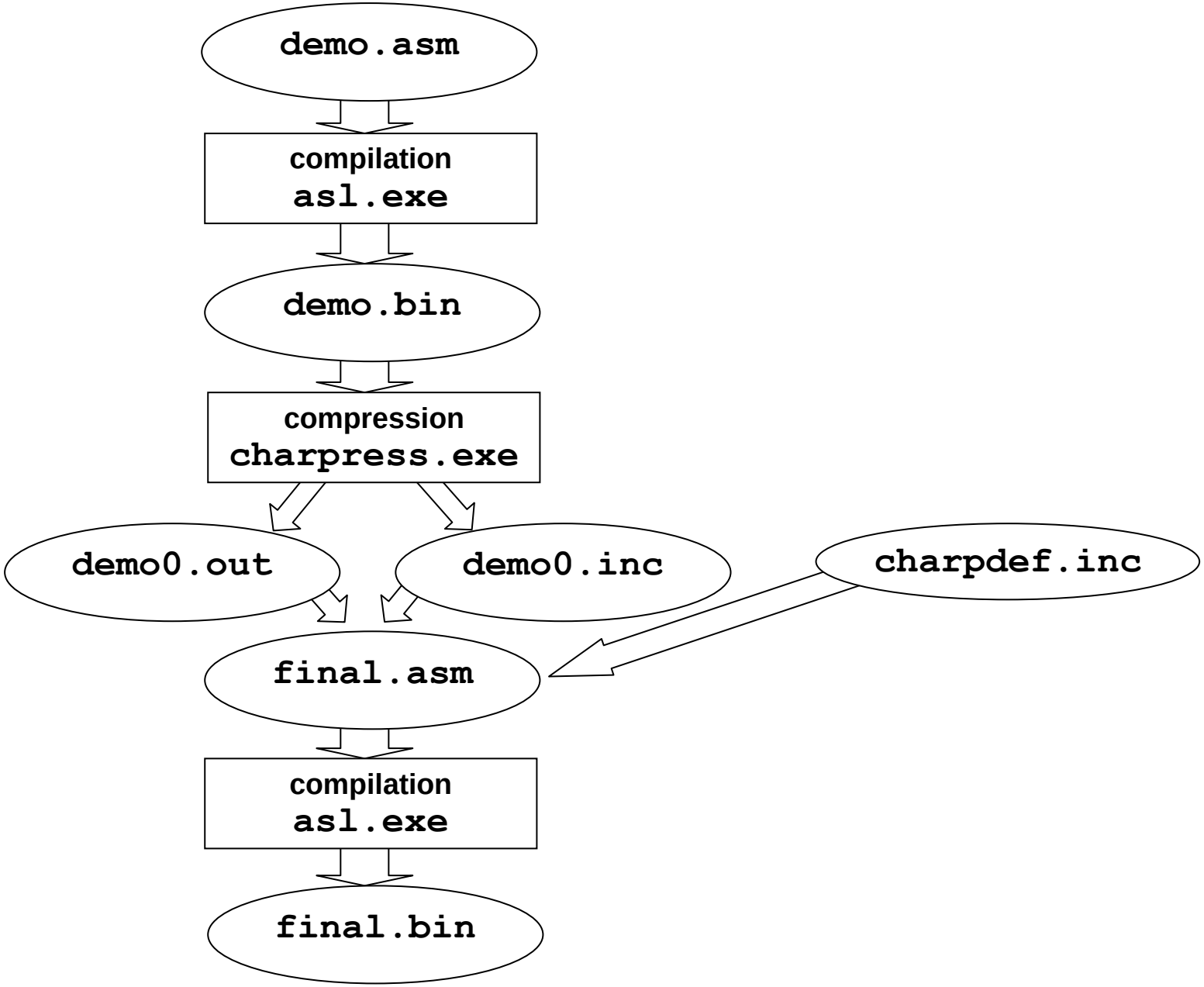
Cíle:

- **Naprostá binární kompatibilita výsledného souboru mezi cross CharPressem a původní ZX verzí**
- **Implementace v „plain C“ zajišťuje další využití kompresoru do budoucna**
- **Maximální modifikovatelnost dekompresní rutiny (rutina již není součástí výstupních dat)**
- **Dávkové zpracování**
- **Dokumentace původního kompresního algoritmu pro Z80**

Výsledky:

- **Všechny cíle byly splněny, výsledky prezentovány na JHCONu 2008**
- **Díky vysoké modifikovatelnosti byl algoritmus využit i pro kompresi dat pro CPC [13]**

Způsob použití Cross CharPress



Literatura

[1] P. Tišnovský : http://www.root.cz/clanky/pcx-prakticky-implementace-komprimace-rle/

[2] Wikipedia RLE: http://cs.wikipedia.org/wiki/RLE

[3] Wikipeda LZ77: http://en.wikipedia.org/wiki/LZ77

[4] Wikipeda LZSS: http://en.wikipedia.org/wiki/LZSS

[5] Wikipedia CZ LZW: http://cs.wikipedia.org/wiki/LZW

[6] Wikipedia CZ Huffman: http://cs.wikipedia.org/wiki/Huffmanovo_k%C3%B3dov%C3%A1n%C3%AD

[7] Wikipedia EN Huffman: http://en.wikipedia.org/wiki/Huffman_coding

[8] Wikipedia Aritmetic coding: http://en.wikipedia.org/wiki/Arithmetic_coding

[10] Žára J., Beneš B., Felkel P.: Moderní počítačová grafika, Computer press 1998, ISBN 80-7226-049-9

[11] P. Biondi, F. Desclaux: Silver Needle in the Skype, EADS CCR

[12] TurboImploder source: http://sysel.webz.cz/system/ti.html

[13] ZeroTeam: cpctro, http://www.pouet.net/prod.php?which=53130