

POT

Semestrální práce

Semestrální práce bude zadána pátý týden, bude se jednat o jednoduchý program v assembleru H8S/2600 v prostředí HEW. Při včasné úspěšné odevzdání semestrální práce dostane student 10 bonusových bodů ke zkoušce.

Test

Test bude ve dvanáctém týdnu a opravný termín bude další týden v hodinách cvičení.

Obsah testu:

- operace s čísly integer a float
- adresní mechanismus paměti
- práce s adresami
- souvislosti mezi adresou a mapou adresního prostoru

zkouška: písemná

Plán výuky:

1. Rezerva.
2. opakování - operace s čísly integer a float.
3. Seznámení s procesorem H8S, úvod do programování v assembleru.
4. Programování v assembleru.
5. Seznámení s prostředím HEW a nástroji GNU. Zadání samostatných prací.
6. Konzultační cvičení - řešení zadaných úloh
7. Konzultační cvičení - řešení zadaných úloh
8. Konzultační cvičení - řešení zadaných úloh
9. Konzultační cvičení - řešení zadaných úloh
10. Strojový cyklus, sběrnice, časování.
11. Dekodéry adres
12. Test.
13. Opravné testy
14. Zápočty

Soustavy užívané v počítačové praxi

Dvojková - binární

$z = 2$ (0,1)

Osmičková - oktalová

$z = 8$ (0,1,2,3,4,5,6,7)

Šestnáctková - hexadecimální

$z = 16$ (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F)

desítkově	dvojkově	osmičkově	šestnáctkově
0	000000	0	0
1	000001	1	1
2	000010	2	2
3	000011	3	3
4	000100	4	4
5	000101	5	5
6	000110	6	6
7	000111	7	7
8	001000	10	8
9	001001	11	9
10	001010	12	A
11	001011	13	B
12	001100	14	C
13	001101	15	D
14	001110	16	E
15	001111	17	F
16	010000	20	10
17	010001	21	11
.			
.			
.			
32	100000	40	20

Zobrazení celého čísla v počítači v binárním tvaru

Pokud nepotřebujeme pracovat s čísly se zápornou hodnotou, ukládáme číslo přímo - nepoužíváme žádné kódování. Rozsah hodnot je $\langle 0-2^n-1 \rangle$ pro $n=8$ dokážeme pracovat s číslem od 0 do 255.

Zobrazení záporných čísel:

přímý kód

rozsah zobrazení $\langle -2^{n-1}+1, 2^{n-1}-1 \rangle$

pro $n=8$ $\langle -127, -0 \rangle, \langle +0, +127 \rangle$

Přímý kód vyčlení první bit znaménku. Pokud například binární číslo 00000001 vyjadřuje jedničku, pak 10000001 označuje -1.

Tento způsob ale komplikuje algoritmy pro praktické počítání – nejprve je vždy třeba testovat znaménkový bit a podle výsledku provést sčítání nebo odčítání. Navíc je třeba mít pro sčítání a odčítání různé algoritmy. Další nevýhodou je, že existují dvě reprezentace čísla nula – kladná nula a záporná nula.

inverzní kód

inverze bitů (jedničkový doplněk)

rozsah zobrazení $\langle -2^{n-1}+1, 2^{n-1}-1 \rangle$

pro $n=8$ $\langle -127, -0 \rangle, \langle +0, +127 \rangle$

Kladná čísla se vyjadřují normálním způsobem, záporná čísla se vyjadřují binární negací čísla.

Například číslo -3 vyjádříme kódem 11111100.

Tento kód má stále dvě reprezentace čísla nula.

doplňkový kód

inverze bitů a přičtení jedničky

rozsah zobrazení $\langle -2^{n-1}, 2^{n-1}-1 \rangle$

pro $n=8$ $\langle -128, 127 \rangle$

Při kódování v doplňkovém kódu je záporné číslo zaznamenáno jako binární negace (záměna všech 0 za 1 a naopak) původního čísla zvětšená o 1. Úvodní bit má v tomto kódu opět význam znaménka. V tomto kódu již existuje jen jediná reprezentace čísla nula, ani není zapotřebí speciální algoritmus pro odčítání.

Příklad:

Vyjádření čísla -13:

převeďme číslo 13 do binární soustavy: 00001101

provedeme negaci: 11110010

přičteme jedničku: 11110011

Sčítání / odčítání:

Pokud se sečte takto vyjádřené záporné číslo s jiným záporným nebo větším kladným číslem, dojde k přetečení rozsahu. Kód je ale zvolen tak, že po odříznutí přetečeného bitu dostáváme správný výsledek.

Kód s posunutou nulou (aditivní kód)

Poslední používanou možností je k číslu připočítat nějakou známou konstantu (třeba $0x7F_{16}=127_{10}$).

Například pro osmibitová čísla, která mohou reprezentovat 256 různých čísel, je možné 00000000 považovat za -127, nulu vyjádříme jako 01111111 a 11111111 je 128.

Nevýhodou tohoto zápisu je, že kladná čísla se liší od bezznaménkové reprezentace čísel. Operace sčítání nepotřebuje úpravy, ale pro operaci násobení je nutné od operandů odečíst známou konstantu.

Tento kód se běžně používá pro reprezentaci exponentu reálných čísel.

Přetečení

Pokud se do paměti snažíme uložit číslo větší, než je daný rozsah, dojde k přetečení. Přetečení se projeví tak, že výsledek dané operace by potřeboval minimálně o jeden bit navíc, který nemáme k dispozici, a proto se do paměti uloží jiné číslo, než jaké by bylo výsledkem operace.

K detekci přetečení máme příznakové bity ve speciálním registru procesoru, jedná se o tzv. status register. Status register může obsahovat různé příznakové bity, ty nejčastější uvádím v tabulce níže.

označení	název	popis
Z	Zero flag	pokud je výsledek operace = 0, nastaví se na 1
C	Carry flag	Pokud došlo k přenosu do vyššího (neexistujícího) bitu při operaci s unsigned číslem, nastaví se na 1
V	Overflow	Pokud došlo k přetečení u signed operace, nastaví se na 1.
N	Negative	Pokud je výsledek operace záporný, nastaví se na 1.
H	Half Carry	Pokud došlo k přenosu z nižších 4 bitů do vyšších, nastaví se na 1.
I	Interrupt Enable	Povolí / zakáže přerušení.

Status register je využit při podmíněných skocích, případně u některých operací (např. rotace a sčítání s využitím carry flag).

Aritmetika ve dvojkových kódech

Sčítání (doplňkový kód)

$20 + (-13) = 00010100 + 11110011 = 1\ 00000111 = 7$ (po odříznutí přeteklého devátého bitu)

```
0 0 0 1 0 1 0 0
1 1 1 1 0 0 1 1
-----
0 0 0 0 0 1 1 1
```

- všechny bity se sčítají stejně (včetně znaménkového)
- vznikne-li přenos ze znaménkového bitu, tak se ignoruje
- přetečení nastane, pokud se přenos do znaménkového bitu nerovná přenosu ze znaménkového bitu

Odčítání (doplňkový kód)

$20 - (-13) = 00010100 - 11110011 = 00100001 = 33$

```
0 0 0 1 0 1 0 0
-1 1 1 1 0 0 1 1
-----
0 0 1 0 0 0 0 1
```

$-76 - 27 = 10110100 - 00011011 = 10011001 = -103$

```
1 0 1 1 0 1 0 0
-0 0 0 1 1 0 1 1
-----
1 0 0 1 1 0 0 1
```

Převod z doplňkového kódu do desítkové soustavy:

Odečteme jedničku a provedeme negaci, pak získané číslo převedeme klasickým způsobem a napíšeme před něj znaménko minus.

$10011001 - 1 = 10011000$

negace... $01100111 = 64 + 32 + 4 + 2 + 1 = 103$

Součet v inverzním kódu

- problém dvou nul
- nutnost provádět tzv. kruhový přenos = přičtení přenosu z nejvyššího řádu k výsledku

Příklad: $-0 + 1$

```
1 1 1 1
0 0 0 1
-----
0 0 0 0
+   1
-----
0 0 0 1
```

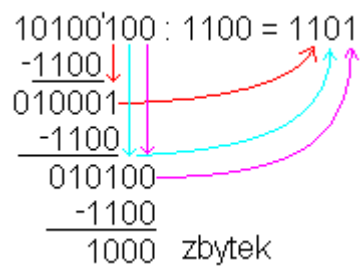
Násobení

$$7 \cdot 5 = 0111 \cdot 0101 = 0111 \cdot 0101 = 100011 = 35$$

```
  0111
* 0101
-----
  0111
 0000
 0111
 0000
-----
100011
```

Dělení

```
10100'100 : 1100 = 1101
-1100
-----
010001
-1100
-----
010100
-1100
-----
1000  zbytek
```



Zobrazení reálného čísla

FORMÁT:

S	Exponent	Mantisa
---	----------	---------

$$+/- (S) \text{ Mantisa} \times 2^{\text{Exponent}}$$

S znaménko čísla 0=kladné, 1=záporné

Mantisa v Přímém kódu (znaménko je v S)

Exponent v kód s posunutou nulou

Normalizovaný tvar Mantisy:

1.xxx

První jedničku (která je tam vždy kromě výsledku 0) vynecháváme.

Příklady:

Př.:

-0.3125₁₀:

$$-0.3125_{10} = -0.0101_2 = -1.01_2 \times 2^{-2}$$

S=1 (-1)

E=-2...=-2+127=125=01111101

Mantisa=.010...

Výsledek:

1 01111101 01000 0

Př.:

1.0

S=0

E=0=127+0=01111111

Mantisa=.0...

Výsledek:

0 01111111 000000 0

Zvláštní čísla podle IEEE 754

0

Exponent i mantisa jsou nulové

Nekonečno:

Exponent obsahuje samé jedničky, a mantisa samé nuly. Znaménkový bit označuje, zda-li jde o kladné, nebo záporné nekonečno.

Nenormalizované číslo:

Při nutnosti zobrazit menší číslo v absolutní hodnotě než je $1.0 \times 2^{-2^{n-1}} + 1$.

O číslu musí být známo, že je nenormalizované, poznáme to tak, že exponent má samé nuly a mantisa je nenulová.

„Not a number“ NaN

Exponent obsahuje samé jedničky a mantisa je nenulová.

Příklady na procvičení

Sečtěte čísla 0x3Eh + 0x4Dh v přímém neznaménkovém kódu a výsledek převed'te do desítkové soustavy.

Převédeme čísla do binární soustavy:

0x3Eh:

3=0011

E=1110 (desítkově 14)

0x3Eh=00111110

0x4Dh:

4=0100

D=1101 (desítkově 13)

0x4Dh=01001101

Napišeme to jako součet pod sebe:

00111110

01001101

10001011

Převédeno do desítkové soustavy:

$1*2^7 + 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 = 128 + 8 + 2 + 1 = \underline{139}$

Stejné hodnoty sečtěte, ale počítejte v doplňkovém kódu

Převod z šestnáctkové soustavy do doplňkového kódu by byl stejný, takže rovnou pod sebe sečteme vstupní čísla.

00111110

01001101

10001011

Převédeno do desítkové soustavy:

První bit je 1, tzn. výsledek je záporný! Převod provádíme tak, že z čísla odečteme jedničku a provedeme negaci:

$10001011 - 1 = 10001010$

$\text{neg}(10001010) = 01110101 = 1*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 = 64 + 32 + 16 + 4 + 1 = 117$

Nakonec ještě připíšeme znaménko!!! Výsledek je -117.

Stejné hodnoty odečtete v přímém neznaménkovém kódu

```
00111110
-01001101
-----
11110001
```

Desítkový výsledek: 241

Z výsledku jasně plyne, že došlo k přetečení (odčítali jsme z kladného čísla větší kladné číslo, výsledek musí být záporný, což neznaménkový kód nedokáže uložit).

Nyní odečtete hodnoty ve znaménkovém kódu

```
00111110
-01001101
-----
11110001
```

Výsledek:

Odečteme jedničku a provedeme negaci:

$11110001 - 1 = 11110000$

$\text{neg}(11110000) = 00001111$

Nezapomeneme na znaménko a výsledek zapíšeme desítkově: -15. K přetečení v tomto případě nedošlo.

Máme čísla 0xA5h a 0x07, řekněte, jakou hodnotu tato čísla představují v přímém neznaménkovém kódu a co představují v doplňkovém kódu.

Převedeme si čísla do binární podoby:

$0xA5h = 10100101$

$0x07h = 00000111$

V přímém kódu by čísla představovala dekadickou hodnotu 165 a 7.

V doplňkovém kódu je číslo 0xA5h -91 a 0x07 je 7.

Takže při sčítání v přímém kódu by mělo vyjít $165 + 7 = 172$ a při sčítání v doplňkovém kódu je to $-91 + 7 = -84$.

Ted' si naše tvrzení ověříme binárně ☺.

```
10100101
00000111
-----
10101100
```

V přímém kódu by to bylo $128 + 32 + 8 + 4 = 172$.

Doplňkový kód: $10101100 - 1 = 10101011$, negace je 01010100, převedeno do desítkové soustavy: -84

Určete, kdy dojde k přetečení:

A	B	operace			
		sčítání		odčítání	
		signed	unsigned	signed	unsigned
50	50	ne	ne	ne	ne
100	50	ano	ne	ne	ne
50	100	ano	ne	ne	ano
150	150	-	ano	-	ne
-50	100	ne	-	ano	-
-50	-100	ano	-	ne	-