ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Semestrální práce z předmětu ÚPA
MIPS

Jméno a příjmení: Martin Sloup
Osobní číslo: A04372
Datum odevzdání: 21. prosince 2006
E-mail: msloup@students.zcu.cz

Zadání

Program převede signed integer na jeho reprezentaci v ASCII, např. 0xB2-> „178“

Nahrazení pseudoinstrukcí

- *Dále musí být v přiloženém výpisu (listing) programu vyznačeny alespoň **tři různé pseudoinstrukce** a musí zde být uvedeno, jakými strojovými instrukcemi je překladač nahradil.*

1) Pseudoinstrukce `move $s0, $ra` je nahrazena `addu $16, $0, $31`
   $16 odpovídá $s0, $0 je vždy nula a $31 odpovídá $ra, tedy do $16 se uloží $0+$31
2) Pseudoinstrukce `li $v0, 4` je nahrazena `ori $2, $0, 4`
   $2 odpovídá $v0, $0 je vždy nula, tedy do $2 ($v0) se uloží $0 or 4
3) Pseudoinstrukce `subu $t2, 1` je nahrazena `addiu $10, $10, -1`
   $10 odpovídá $t2, tedy $10 = $10 + (-1)

Datový hazard

- Kromě toho musí být v programu vyznačeno alespoň jedno místo, kde potenciálně vzniká **datový hazard** a jedno místo, kde se potenciálně může projevit (resp. kde se projevuje) **zpožděné čtení dat z paměti**.

Potenciální datový hazard nastává v případech, kdy se instrukce pokusí přečíst obsah paměti po instrukci, která do tohoto zdroje zapisuje. Bohužel jsem ve svém kódu nenalezl potenciální datový hazard, ale mohu alespoň ukázat, při jakém případu by nastalo.

Např. po prohozeni dvou instrukcí po načítaní ze zásobníku, v mém případě, vzniká tato situace:

```
addi $sp, $sp, 1
lb $t0, ($sp)
```

Zpožděné čtení dat z paměti:
Zpožděné čtení dat z paměti nastane za situace, kdy načítám data z paměti a ihned v další instrukci s těmito daty dále pracuji. Ani tuto situaci jsem v programu nenašel, ale pro ukázku uvedu příklad, kdy to nastane po odstranění pár instrukcí:

```
lb $t0, ($sp)
beqz $t0, PLOOP
```

Výpis programu

```
#  (C) 2006 Martin Sloup, A04372
#  E-mail: msloup@students.zcu.cz
#  Vytvoreno v ramci semestralni prace z KIV/UPA
#  Kod prevadi Signed integer na jeho reprezentaci v ASCII, napr 0xB2 -> "178"

  .data            # datovy segment
OVSTUP:    .asciiz  "Zadej cislo: "      #vstupni textova hlaska
ODPOVED:   .asciiz  "Cislo v ASCII: "    #vystupni textova hlaska

  .text
  .globl main

    # deleni 10, zbytek se ulozi do $v1, quocient do $v0
DELENI:
    #dvou parametrove div mi nechtelo fungovat, tak se to muselo napsat takhle:
```

```
    div      $v0, $a0, 10      # provedeme deleni deseti, v $v1 je zbytek, v $v0
quocient
    mul      $t0, $v0, 10
    sub      $v1, $a0, $t0

    jr       $ra               #vratime se zpet do hlavniho programu
    nop

main:
    move     $s0,  $ra         #ulozime si navratovou pozici programu

     # tiskni "Zadej cislo:"
    li       $v0, 4
    la       $a0, OVSTUP
    syscall

    #vstup cisla z klavesnice
    li       $v0, 5
    syscall

    #hodime cislo z vystupu do $t0
    move     $t0, $v0

    #pamatovak na znamenko nastavime na 0
    li       $t1, 0
    #nejprve zjistime znamenko
    bgezal   $t0, BEZZN        # if $t0 >= 0 then goto BEZZN
    nop

    #pokud je cislo zaporne, provedeme negaci a pricteni jednicky
    not      $t0, $t0
    add      $t0, $t0, 1

    #ulozime pamatovak na znamenko
    li       $t1, 1
BEZZN:
    #promenou na cyklus nastavime na 5
    li       $t2, 5
    nop

DALD:
    #pokud probehl cyklus 5x skocime na PODEL
    blez     $t2, PODEL
    nop

    #provedeme subrutinu deleni, jako parametr vstpuje delenec
    move     $a0, $t0
    jal      DELENI           # deleni
    nop
    #jako vystup vyleza $v0 pro quocient a $v1 jako zbytek

    #ulozime zbytek do zasobniku
    addi     $sp, $sp, -1
    sb       $v1, ($sp)
    nop

    #do $t0 nastavime quocient
    move     $t0, $v0

    #snizime pocitadlo cyklu
    subu     $t2, 1

    # a provedem dalsi cyklus
    j        DALD
    nop
```

```
PODEL:

    #vyhodime text odpovedi
    li       $v0, 4
    la       $a0, ODPOVED
    syscall

# predelani na text
    # nejprve znamenko, pokud bylo zadane cislo zaporne
    beqz     $t1, BTEXTZN
    nop

    #zobrazime minus
    # misto 11 napsat jen 1, pokud vypisovat jako integer
    li       $v0, 11
    li       $a0, '-'
    syscall

    #odkomentovat, pokud psat carku
#    li       $v0, 11
#    li       $a0, ','
#    syscall

BTEXTZN:
    #promenou na cyklus nastavime na 5
    li       $t2, 5
    #pamatovak, zda uz bylo nulove cislo nastavime na 0
    li       $t1, 0

PLOOP:
    #pokud cyklus probehl 5x skocime na ONULY
    blez     $t2, ONULY
    nop

    #naceteme cislo ze zasobniku
    lb       $t0, ($sp)
    addi     $sp, $sp, 1
    nop

    #snizime cislo cyklu
    subu     $t2, 1

    #pokud jiz bylo cislo nenulove, skocime na tisk cisla
    bgtz     $t1, TISK
    nop

    #je-li cislo nulove pokracujeme na dalsi
    beqz     $t0, PLOOP
    nop
    #je-li nenulove, nastavime pamatovak, ze se tak stalo
    li       $t1, 1

TISK:
    #tisk cisla
    #pricetem k cislu ascii hodnotu znaku nuly
    add      $t0, $t0, '0'

    #vypiseme na obrazovku
    # misto 11 napsat jen 1, pokud vypisovat jako integer
    li       $v0, 11
    move     $a0, $t0
    syscall

    #odkomentovat, pokud psat carku
#    li       $v0, 11
```

```
#    li       $a0, ','
#    syscall


    #skocime na dalsi cislo
    j        PLOOP
    nop

ONULY:
    #nyni osetrime, pokud bylo zadano nulove cislo, to pozname tak,
    #ze se z haldy nacetla sama nulova cisla
    bnez     $t1, KONEC
    nop

    # misto 11 napsat jen 1, pokud vypisovat jako integer
    li       $v0, 11
    li       $a0, '0'
    syscall

KONEC:
    #novy radek na konec
    li       $v0, 11
    li       $a0, 0x0A
    syscall

    #nastavime zpatky $ra
    move     $ra, $s0
    jr       $ra
    nop
```

Vstup a výstup programu

Veškeré vstupy a výstupy do programu se provádí prostřednictvím systémových volání. A následně uvádím i výstup kdy se nastaví výpis výstupu znaku z char na integer (okomentováno v kódu):

```
Zadej cislo: 14562
Cislo v ASCII: 14562
Cislo v ASCII: 49,52,53,54,50,
```

Příklad pro záporná čísla:

```
Zadej cislo: -1382
Cislo v ASCII: -1382
Cislo v ASCII: 45,49,51,56,50,
```

Závěr

K zapsání zdrojového kódu programu jsem použil freewarový editor PSPad a k odladění a simulaci běhu programu byl použit program PCSpim ve verzi 7.2.1.
Úloha jsem vyřešil podle zadání tak, aby program nebyl ovlivněn zpožděním instrukcí load a skoků.
Během psaní programu jsem se seznámil s architekturou MIPS, s používáním pseudoinstrukcí a s řešením problémů se zpožděním instrukcí pro load a skoky.