Musterlösung Blatt 6: Assembler-Programmierung 2

Aufgabe 1: Puzzel

1 Was macht das Programm?

a) Das Programm berechnet, ob die Zahl NUM eine Primzahl ist:

$$\forall i \in \{2, \ldots, NUM - 1\} : NUM \bmod i \neq 0 \Rightarrow NUM \text{ ist Prim}$$

Damit das Programm korrekt arbeitet muss gelten: $NUM > 3$

b) Der Programm-Ablaufplan sieht folgendermßen aus:

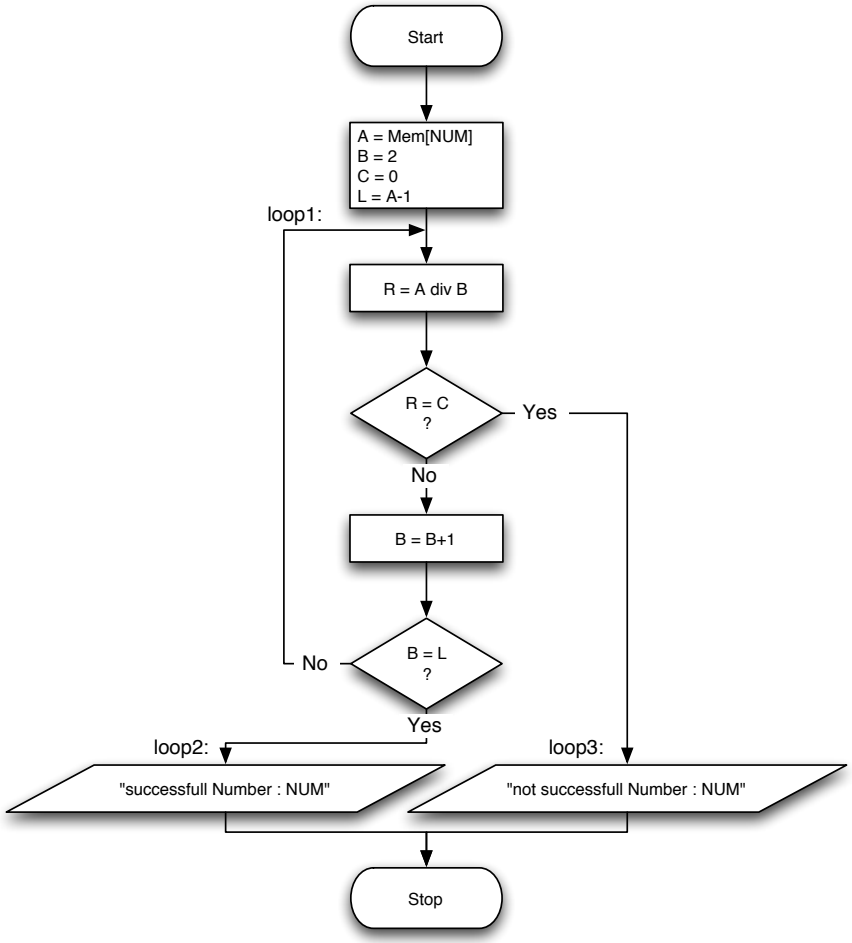

Abbildung 1: Programmablaufplan des Assembler-Programms „Puzzel“

Aufgabe 2: Erweiterung

a) Folgende Schritte sind notwendig um einen Funktionsaufruf durchzuführen:

1) gegebenenfalls Platz auf dem Stack schaffen, um die Parameter und *Caller-Saved Register* zu sichern
2) Übergabe der Parameter: folgende Konvention gilt: die ersten vier Parameter werden in den Registern $a0 – $a3 übergeben, der Rest auf dem Stack.
3) Sichern der eigenen temporären Register $t0 – $t9, falls deren Werte nach dem Aufruf noch benötigt werden
4) Ausführen eines Sprungs mit Sichern der Rüchsprungadresse in $ar

Am Anfang des Unterprogramms sind folgende Schritte durchzuführen:

1) Einrichten des *Stack Frames* durch Substraktion seiner Grösse vom *Stack Pointer*
2) Sichern des *Frame Pointer* auf dem Stack
3) Sichern der Rücksprungadresse in $ar, falls das Unterprogramm ebenfalls Unterprogramme aufruft
4) Sichern der Register $s0 – $s7, falls diese in dem Unterprogramm benötigt werden
5) Setzen des *Frame Pointer* durch Addition der Frame-Grösse auf den *Stack Pointer*

Am Ende des Unterprogramms sind folgende Schritte nötig:

1) Wiederherstellung aller gesicherten Register ($fp, $ar, $s0 – $s7)
2) Löschen des *Stack Frame* durch Subtraktion seiner Grösse vom *Stack Pointer*
3) Rücksprung an die Adresse im Register $ar

Folgende Schritte sind nach dem Unterprogrammaufruf im aufrufenden Programm nötig:

1) eventuell Wiederherstellung der gesicherten Register ($t0 – $t9)
2) gegebenenfalls den Stack wieder verkleinern
3) eventuell Sichern des Funktionsergebnisses ($v0)

Hinweis: Der *Stack Frame* beinhaltet alle gesicherten Register und alle lokalen Variablen der Funktion, soweit diese nicht alle in die Register passen. Der Framepointer zeigt auf die erste Speicherstelle unter den Parametern auf dem Stack und damit auf das erste Byte der *Callee-Saved Registers*. Der Stackpointer zeigt immer auf die letzte benutzte Speicherzelle des Stack.

b) Das Assebler-Programm mit der neuen *div*-Unterroutine so sieht folgendermaßen aus:

```
        ...

        lw      $a1, NUM
        li      $s2, 2
        addi    $s4, $a1, -1    # 0 NUM-1
                ^^^             # s4 instead of t2 (backup of t2
                                # is not necessary anymore)
        move    $s3, $a1        # backup NUM
loop1:  move    $a1, $s3        # set parameter 1
        move    $a2, $s2        # set parameter 2
                                # for more parameters the stack must be
                                #  adjusted accordingly (not required here)
                                # saving of caller registers $t0-$t9 not
                                # required
        jal     sub_div         # jump to subroutine
                                # restoring of caller registers $t0-$t9 not
                                # required
                                # adjust stack pointer iff required (here
                                # not)
        move    $t3, $v0        # get result
        beq     $t3, $ZERO, loop3
        addi    $s2, $s2, 1
        beq     $s2, $s4, loop2
        j       loop1
        ...

# The new subroutine starts here
sub_div: sub    $sp, $sp, 4     # create stack frame for saving registers
        sw      $fp, 4($sp)     # save $fp
        move    $fp, $sp
        add     $fp, $fp, 4     # set frame pointer accordingly
loop4:  sub     $a1, $a1, $a2
        bgez    $a1, loop4
        add     $a1, $a1, $a2
        move    $v0, $a1        # set result
        lw      $fp, 4($sp)     # restore $fp
        add     $sp, $sp, 4     # adjust stack pointer
        jr      $ra             # return
```