Testat 1

Einführung in die Systemprogrammierung SS 2014

27. Mai 2014

Abgabedatum: 20:00, 03.06.2014

In diesem Testat behandeln wir die nebenläufige Ausführung von Programmen im Kontext des MIPS-R2000-Prozessors mit einem Kern.

Abgabemodalitäten

Das Ergebnisprogramm für dieses Testat muß in das WebSPIM-System eingetragen und eingesendet werden. Es gelten folgende Regeln zur Bearbeitung, Abgabe und Bewertung:

a. Sie können die Aufgabe alleine oder zu zweit bearbeiten und einsenden.

b. Sie müssen bei der Abgabe Ihre Namen, Matrikelnummern, und E-Mailadressen angeben (z.B. als Kommentare im Quellcode, oder im Kommentarfeld).

c. Der eingesendete Code muß vollständig von Ihrem Team entwickelt worden sein.

d. Jedes Teammitglied muß in der Lage sein, den Code eigenständig zu erklären.

e. Wir behalten uns vor, Punktvergabe von persönlichen Einzelgesprächen abhängig zu machen.

f. Sie dürfen sich mit anderen Studierenden austauschen. Vermeiden Sie dabei das direkte Zeigen von Codefragmenten.

1 Aufgabenstellung

Gegeben sind vier MIPS-Teilprogramme mit Einstiegspunkten `main0`, `main1`, `main2` und `main3`. Diese Teilprogramme verwenden einen bestimmten, vorgegebenen Befehlssatz (s.u.).

Schreiben Sie folgendes Hauptprogramm:

- Ihr Programm soll diese vier Teilprogramme „nebeneinander“ ausführen, indem es jeweils einen Maschinenbefehl jedes Teilprogrammes ausführt und dann zum nächsten Teilprogramm wechselt.
- Die Teilprogramme sollen im ‘Round-Robin’-Verfahren ausgeführt werden, also erst `main0`, dann `main1`, dann `main2`, dann `main3`, und dann wieder `main0` usw.
- Teilprogramme können sich durch `syscall` 10 beenden. In diesem Fall soll nur das angegebene Teilprogramm beendet (also beim Round-Robin übersprungen) werden. Andere `syscall`-Aufrufe sollen wie üblich erlaubt bleiben.
- Wenn ein Teilprogramm beendet wurde, geben Sie „Teilprogramm x beendet“ aus, wobei ‘x’ die Nummer (0–3) des Teilprogrammes ist.
- Wenn alle Teilprogramme beendet wurden, beenden Sie die Ausführung Ihres Hauptprogrammes nach der Meldung „Ende“.
- Sie dürfen die vier gegebenen Programme nicht verändern.

- Ihr Programm muß auch andere Teilprogramme als die vier im WebSPIM angegebenen nebeneinander auszuführen können, sofern diese Teilprogramme sich auf den vorgegebenen Befehlssatz beschränken.
- Sie können davon ausgehen, daß der Ausfühungsstapel nie mehr als 1 kiB an Speicher benötigt.
- 20% der Punktzahl hängen davon ab, daß Sie den kompletten angegebenen Befehlssatz unterstützen, auch, wenn er nicht von den angegebenen Teilprogrammen verwendet wird.

Befehle

1.1 R-Befehle

Assembler	**6**	**5**	**5**	**5**	**5**	**6**
`add rd, rs, rt`	0	rs	rt	rd	0	0x20
`addu rd, rs, rt`	0	rs	rt	rd	0	0x21
`and rd, rs, rt`	0	rs	rt	rd	0	0x24
`div rs, rt`	0	rs	rt	0	0	0x1a
`divu rs, rt`	0	rs	rt	0	0	0x1b
`mult rs, rt`	0	rs	rt	0	0	0x18
`multu rs, rt`	0	rs	rt	0	0	0x19
`nor rd, rs, rt`	0	rs	rt	rd	0	0x27
`or rd, rs, rt`	0	rs	rt	rd	0	0x25
`sll rd, rt,` s	0	rs	rt	rd	s	0
`sllv rd, rt, rs`	0	rs	rt	rd	0	0x04
`sra rd, rt,` s	0	rs	rt	rd	s	0x03
`srav rd, rt, rs`	0	rs	rt	rd	0	0x07
`srl rd, rt,` s	0	rs	rt	rd	s	0x02
`srlv rd, rt, rs`	0	rs	rt	rd	0	0x06
`sub rd, rs, rt`	0	rs	rt	rd	0	0x22
`subu rd, rs, rt`	0	rs	rt	rd	0	0x23
`xor rd, rs, rt`	0	rs	rt	rd	0	0x26
`slt rd, rs, rt`	0	rs	rt	rd	0	0x2a
`sltu rd, rs, rt`	0	rs	rt	rd	0	0x2b
`mfhi rd`	0	0	0	rd	0	0x10
`mflo rd`	0	0	0	rd	0	0x12
`mthi rs`	0	rs	0	0	0	0x11
`mtlo rs`	0	rs	0	0	0	0x13
`mul rd, rs, rt`	0x1c	rs	rt	rd	0	0x02

1.2 I-Befehle

Assembler	**6**	**5**	**5**	**16**
`addi rt, rs, Direktwert`	0x08	rs	rt	*Direktwert*
`addiu rt, rs, Direktwert`	0x09	rs	rt	*Direktwert*
`andi rt, rs, Direktwert`	0x0c	rs	rt	*Direktwert*
`ori rt, rs, Direktwert`	0x0d	rs	rt	*Direktwert*
`xori rt, rs, Direktwert`	0x0e	rs	rt	*Direktwert*
`lui rt, Direktwert`	0x0f	0	rt	*Direktwert*
`slti rt, rs, Direktwert`	0x0a	rs	rt	*Direktwert*
`sltiu rt, rs, Direktwert`	0x0b	rs	rt	*Direktwert*
`beq rs, rt, Sprungmarke`	0x04	rs	rt	*Abstand*
`bgez rs, Sprungmarke`	0x01	rs	0x01	*Abstand*
`bgezal rs, Sprungmarke`	0x01	rs	0x11	*Abstand*
`bgtz rs, Sprungmarke`	0x07	rs	0	*Abstand*
`blez rs, Sprungmarke`	0x06	rs	0	*Abstand*
`bltz rs, Sprungmarke`	0x01	rs	0	*Abstand*
`bne rs, rt, Sprungmarke`	0x05	rs	rt	*Abstand*
`lb rt, Abstand(rs)`	0x20	rs	rt	*Abstand*
`lbu rt, Abstand(rs)`	0x24	rs	rt	*Abstand*
`lh rt, Abstand(rs)`	0x21	rs	rt	*Abstand*
`lhu rt, Abstand(rs)`	0x25	rs	rt	*Abstand*
`lw rt, Abstand(rs)`	0x23	rs	rt	*Abstand*
`sb rt, Abstand(rs)`	0x28	rs	rt	*Abstand*
`sh rt, Abstand(rs)`	0x29	rs	rt	*Abstand*
`sw rt, Abstand(rs)`	0x2b	rs	rt	*Abstand*

Dekodierung Abstand zu Sprungzieladresse:

- 2 Bits nach links schieben
- Vorzeichen erweitern (soweit nötig)
- Auf Programmzähler aufaddieren *statt +4*

1.3 J-Befehle

Assembler	**6**	**26**
`j Ziel`	0x02	*Ziel*
`jal Ziel`	0x03	*Ziel*

Dekodierung der Zieladressen: 2 Bits nach Links schieben