

INSTITUT FÜR INFORMATIK
Lehrstuhl für Rechnerarchitektur (Informatik 3)
Universität Erlangen-Nürnberg
Martensstr. 3, 91058 Erlangen

2.10.2003

Klausur
 zu

"Organisation und Technologie von Rechensystemen 4"

.....
 Matrikelnummer Geb.-Datum Vorname Name

- Es sind keine elektronischen Hilfsmittel erlaubt.
- Legen Sie den Ausweis (mit Lichtbild!) griffbereit auf den Platz.
- Dieses Aufgabenheft umfaßt 12 Seiten. Überprüfen Sie die Vollständigkeit.
- Gesondert beigelegte Blätter werden nicht bewertet!
- Schreiben Sie deutlich! Unleserliches wird nicht bewertet!
- Es darf nicht mit der Farbe rot geschrieben werden!
- Bei Bestehen wird der Schein direkt an das Prüfungsamt geschickt!

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausurunterlagen
- die Kenntnisnahme der obigen Informationen.

Erlangen, den 2.10.2003

(Unterschrift)

Ich bin damit einverstanden, daß mein Prüfungsergebnis der Klausur zu OTRS 4 unter Angabe der Matrikel-Nummer veröffentlicht wird.

Erlangen, den 2.10.2003

(Unterschrift)

Aufgabe	1	2	3	4	5	Summe
max. Punktzahl	5	8	11	16	20	60
erreichte Punktzahl						

Aufgabe 1: Allgemeines**(5 Punkte)**

(Je Teilaufgabe 1 Punkt; Punktabzug bei falschen Antworten!)

1a) Was ist ein Program Counter (PC)?

- Register, in dem die Anzahl der Programme auf einem Rechner gezählt wird
- Register, das die Adresse des aktuellen Befehls enthält
- Schaltwerk, das die abgelaufenen Schritte eines Mikroprogramms zählt
- Speicherstelle, in der unter UNIX die Anzahl der gleichzeitig laufenden Kopien einer Programmdatei festgehalten wird

1b) Wofür wird Platz auf dem Stack verwendet?

- globale Variablen
- Rücksprungadressen
- lokale Variablen
- Bildschirmspeicher

1c) "Big Endian" und "Little Endian"

- Bei Little Endian steht das höchstwertige Byte an der höchsten Adresse
- Grosse Zahlen speichert man im Big Endian-Format, kleine im Little Endian
- Bei 8-Bit-Prozessoren nennt man das Speicherformat "Little Endian", bei 32-Bit-Prozessoren "Big Endian", um die Rechnerarten im Netz unterscheiden zu können
- Das Big Endian Format braucht genauso viel Speicher wie das Little Endian Format

1d) Wo liegt ein Stackframe?

- Auf dem Stack
- Auf dem Heap
- Im entsprechenden Prozessorregister
- Normalerweise auf der Festplatte, bei Platten mit mehreren Oberflächen (Stapel)

1e) Wozu benutzt man im Normalfall Prozessorflags ?

- Erkennung bestimmter Zustände oder Rechenergebnisse
- Ausführen bedingter Verzweigungen
- Verhindern/Erlauben von Interrupts
- Generierung von Bitmustern für Bildschirmspeicher

Aufgabe 2: Speicher**(8 Punkte)**

Gegeben sei die folgende Datenstruktur eines 32-Bit-Rechners:

```
struct data {
    char name[14];
    int note;
    int punkte;
};
```

Ein Datensatz gemäß dieser Struktur sei mit korrektem Alignment ab Adresse 1240 (dezimal) gespeichert.

Das folgende ist ein Speicherauszug des Speichers von Adresse 1232 (dezimal) bis 1280 (dezimal). Die Werte der Bytes dieses Bereiches werden im 16er-System angegeben:

```
1216: ...
1232: 00 00 00 01 00 00 00 40 4D 45 49 45 52 00 00 00
1248: 00 00 00 00 00 00 6A 12 00 00 00 03 00 00 00 28
1264: 4D 55 45 4C 4C 45 52 00 00 00 00 00 00 00 00 00
1280: ...
```

Wie heisst die/der Student(in) (name), deren/dessen Daten in diesem Bereich abgespeichert sind? Wieviele Punkte hat sie/er in der Klausur erreicht (punkte) und welche Note gab es dafür (note)? In welchem Format (Big-Endian, Little-Endian) speichert der Rechner vermutlich die Daten? Begründen Sie Ihre Antworten!

Hinweis: der Name ist im ASCII-Format abgespeichert ('A' = 0x41, 'B' = 0x42, ..., 'E' = 0x45, ..., 'I' = 0x49, ..., 'L' = 0x4C, ..., 'R' = 0x52, ..., 'U' = 0x55, ... 'Z' = 0x5A).

Aufgabe 3: Kontrollstrukturen**(11 Punkte)**

Gegeben sei folgender Assembler-Code:

```
func:
    movl 4(%esp),%eax
    cmpl $15,%eax
    ja L23
    shll $2, %eax
    movl L24(%eax), %eax
    jmp *%eax
L24:
    .long L4
    .long L5
    .
    .
    .long L18
    .long L19
L4:
    call func0
    jmp L3
L5:
    call func1
    jmp L3
    ...
L18:
    call func14
    jmp L3
L19:
    call func15
    jmp L3
L23:
    jmp L23
L3:
    ret
```

Welchen Hochsprachen-Code könnte ein Compiler hier compiliert haben?

(Hinweis: `jmp *%eax` springt zu der Anweisung, deren Adresse in `%eax` steht.)

Aufgabe 4: Stack / Unterprogramm**(16 Punkte)**

Gegeben sei folgendes Unterprogramm:

```
void
proc(void)
{
    unsigned long res;
    unsigned long x;
    unsigned long y;
    unsigned long z;

    x = inl(0x3f0);
    y = inl(0x3f4);
    z = inl(0x3f8);
    res = x + y + z;
    outl(res, 0x3fc);
}
```

`inl` und `outl` sind externe Funktionen mit Standard-Parameter- und -Return-Wert-Übergabe. Wie sie implementiert sind, sei nicht bekannt.

- 4a) Schreiben Sie ein semantisch äquivalentes Hochsprachenprogramm, das möglichst wenig Platz auf dem Stack verwendet! Gehen Sie davon aus, dass jedes Unterprogramm den Wert von zwei Registern der CPU (z.B. `%eax` und `%edx`) ändern darf! (6 Punkte)

4b) Compilieren Sie das Unterprogramm! Kommentieren Sie Ihren Code! (6 Punkte)

4c) Was müsste an Ihrem Code geändert werden, wenn dieses Unterprogramm als Interrupt-Prozedur verwendet werden sollte? Begründen Sie Ihre Antwort! (4 Punkte)

Aufgabe 5: Mikrocontroller / Hardware**(20 Punkte)**

Ein Sensor liefere ein Längenmaß Z in $1/64''$ -Einheiten als 6-Bit Binärzahl (0..63).

$1'' = 25.4\text{mm}$ ($1'' = 1\text{ inch} = 1\text{ engl. Zoll}$)

Ausgegeben werden soll das Maß als Byte M (0..250) in $1/10\text{ mm}$ -Einheiten, mit einem

Rundungsfehler $\leq 1/2$ Einheit.

Daraus ergibt sich folgende zu implementierende Funktion:

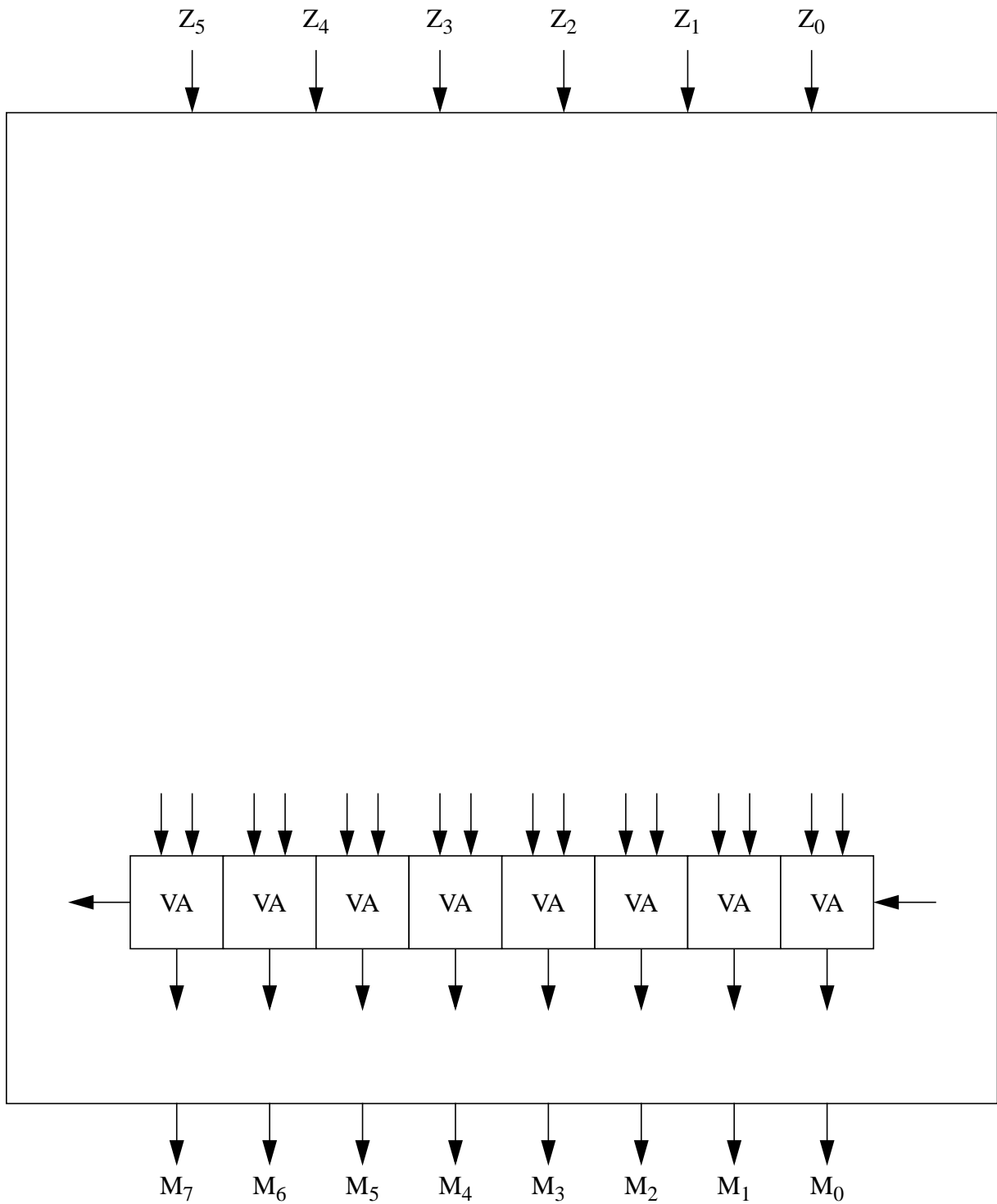
```
unsigned int
convert(unsigned int Z)
{
    unsigned int M;

    M = (Z * 254 + 31) / 64;

    return M;
}
```

- 5a) Schreiben Sie ein Assembler-Unterprogramm, das die Umrechnung vornimmt! Es dürfen keine Multiplikations- oder Divisionsbefehle verwendet werden! (10 Punkte)

5b) Geben Sie eine Hardware-Schaltung zur Umwandlung $Z \rightarrow M$ an! Es stehen ein 8-Bit-Volladdierer (VA) und die gängigen Logikelemente zur Verfügung. (10 Punkte)



Zusätzlicher Platz

Zusätzlicher Platz

Zusätzlicher Platz