1709 ML NK 03

## Aufgabe 1: Cache (1+2+3+3+3+2= 14 Punkte)

Gegeben Sei ein Prozessor mit Byte-Adressierung, bei dem der Cache-Speicher bereits auf dem Prozessor-Chip integriert ist. Die Organisationsparameter dieses Cache-Speichers führen zu der folgenden Aufteilung der Zugriffsadresse in die Felder Tag, Index und Byte:

| Zugriffsadresse | | | | |
|---|---|---|---|---|
| MSB | Tag | Index | Byte | LSB |
| | 14 Bit | 13 Bit | 5 Bit | |

a) Wie groß ist der unterstützte Adressraum des Prozessors in MByte oder GByte?

b) Der Cache-Speicher sei 16-fach set-assoziativ ausgelegt. Welche Speicherkapazität hat der Inhaltsteil des Cache-Speichers (also ohne Tags und Status-Flags?)

c) Wie viele 1-Bit-Vergleicher (Äquivalenzgatter) werden für den gleichzeitigen Vergleich der Tags benötigt?

d) Geben Sie in der folgenden Tabelle an, wie sich die Speicherkapazität und der unterstützte Adressraum des Cache-Speichers ändern („kleiner", „gleich" oder „größer"), wenn einzelne Organisationsparameter geändert werden. Dabei bezieht sich jede Änderung auf die Aufgabenteil a) beschriebene Ausgangssituation.

| Tag | Index | Byte | Assoziativität | Cache-Kapazität | Adressraum |
|---|---|---|---|---|---|
| 14 Bit | 13 Bit | 5 Bit | 16-fach | Ausgangssituation | Ausgangssituation |
| **15 Bit** | 13 Bit | 5 Bit | 16-fach | | |
| 14 Bit | **14 Bit** | 5 Bit | 16-fach | | |
| 14 Bit | 13 Bit | **6 Bit** | 16-fach | | |
| 14 Bit | 13 Bit | 5 Bit | **32-fach** | | |

e) Der Prozessorhersteller möchte den Prozessor so weiterentwickeln, dass er einen doppelt so großen Adressraum unterstützt (ein Adressbit mehr). Aus Kostengründen soll die Speicherkapazität des Cache-Speichers (Inhaltsteil) jedoch unverändert bleiben. Nennen Sie drei Vorgehensweisen, wie Organisationsparameter „Tag", "Index", „Byte" und Assoziativität geändert werden können, um dieses Ziel zu erreichen.

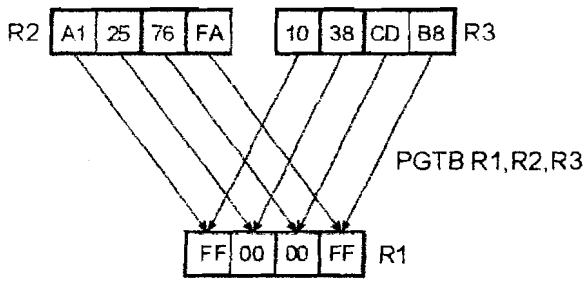
f) Bei welche(r/n) Maßnahme(n) aus Teilaufgabe e) ist zu erwarten, dass die vorhandenen Programme zu derselben Trefferrate des Cache-Speichers führen, die auch die Vorgängerversion des Prozessors erzielt hat? Begründen Sie Ihre Antwort!

## Aufgabe 2: MMX (Multi Media eXtension) (10+6= 16 Punkte)

In dieser Aufgabe verwenden wir ein paar vereinfachte MMX-Befehle, deren Wirkungsweise nun erklärt wird:

| | | |
|---|---|---|
| PAND | R1,R2,R3 | $R1 = R2 \wedge R3$ |
| POR | R1,R2,R3 | $R1 = R2 \vee R3$ |
| PNOT | R1,R2 | R1 = NOT R2 |
| PGTB | R1,R2,R3 | $R1 \leftarrow R2 > R3$ |

PGTB bedeutet *Parallel Greater Than Byte*. Wenn ein 8-Bit Paket in Register R2 (vorzeichenlos) größer ist als das zugehörige Byte in Register R3, wird in Register R1 $FF abgelegt, ansonsten $00.




a) Erstellen Sie ein kurzes Programm mit den oben angegebenen MMX-Befehlen zur Berechnung der byteweisen Maxima in R2 und R3 und legen Sie das Ergebnis in R1 ab. Dazu stehen Ihnen die Register R1 bis R15 zur Verfügung. Nummerieren Sie jeden Befehl. Verwenden Sie nur die genannten Befehle.

b) Prüfen Sie Ihr Programm mit folgenden Registerinhalten:

**R2= $A12576FA, R3=$1038CDB8.**

Schreiben Sie hinter jeden Befehl Ihres Programms die Registerinhalte der veränderten Register. Gehen Sie davon aus, das R2 und R3 bereits die genannten Werte enthalten.

*Musterlösung:*

*a)*

| | | | |
|---|---|---|---|
| 1 | PGTB | R4,R2,R3 | *R4 enthält Maske für R2>R3* |
| 2 | PAND | R1,R2,R4 | *R1 : Bytes von R2 mit R2>R3 sonst 0* |
| 3 | PNOT | R4,R4 | *R4 : Maske R3 > R2* |
| 4 | PAND | R2,R4,R3 | *R2 : Bytes von R3 mit R3>R2, sonst 0* |
| 5 | POR | R1,R1,R2 | *R1 enthält max(R2,R3) byteweise* |

*b)*

| | | | |
|---|---|---|---|
| 1 | PGTB | R4,R2,R3 | *R4 = $FF 00 00 FF* |
| 2 | PAND | R1,R2,R4 | *R1 = $A1 00 00 FA* |
| 3 | PNOT | R4,R4 | *R4 = $00 FF FF 00* |
| 4 | PAND | R2,R4,R3 | *R2 = $00 38 CD 00* |
| 5 | POR | R1,R1,R2 | *R1 = $A1 38 CD FA* |

## Aufgabe 3: Gemeinsamer Ein-Bit-Prädiktor: (3+3+2+2= 10 Punkte)

Gegeben sei ein Programm, bei dem zwei Verzweigungsbefehle *B1* und *B2* alternierend ausgeführt werden. Da die Verzweigungsadressen in den niederwertigen Bits übereinstimmen, werden sie in der Sprungverlaufstabelle auf die gleiche Speicherzelle abgebildet. *B1* und *B2* teilen sich somit einen gemeinsamen Prädiktor.

a) Bestimmen Sie die Vorhersagegüte, wenn das Programm folgenden Verlauf aufweist (T=branch Taken; NT branch Not Taken):

| B1 | B2 | B1 | B2 | B1 | B2 | B1 | B2 |
|---|---|---|---|---|---|---|---|
| T | NT | NT | T | T | NT | NT | T |

Der Ein-Bit-Prädiktor werde mit NT initialisiert.

b) Welche Vorhersagegüte ergäbe sich, wenn die Verzweigungsadressen nicht in den niederwertigen Bits übereinstimmen würden? In diesem Fall gibt es zwei voneinander unabhängige Ein-Bit-Prädiktoren. Die beiden Prädiktoren werden jeweils mit NT initialisiert.

c) Konstruieren Sie ein Verlaufsmuster, das mit einem gemeinsamen Prädiktor stets falsche Vorhersagen liefert!

d) Erläutern Sie, weshalb gemeinsame Prädiktoren bei langen Befehlsfolgen realer Programme gegenüber getrennten Prädiktoren meist eine geringere Vorhersagegüte aufweisen.

## *Musterlösung:*

a) Es ergibt sich eine Vorhersagegüte von 3/8 oder 37,5 %.

| | B1 | B2 | B1 | B2 | B1 | B2 | B1 | B2 |
|---|---|---|---|---|---|---|---|---|
| | T | NT | NT | T | T | NT | NT | T |
| Vorhersage korrekt | Nein | Nein | Ja | Nein | Ja | Nein | Ja | Nein |

b) Es ergibt sich eine Vorhersagegüte von 1/8 oder 12,5 %. Erstaunlicherweise verschlechtert sich die Vorhersagegüte durch den Einsatz getrennter Prädiktoren.

| | B1 | B2 | B1 | B2 | B1 | B2 | B1 | B2 |
|---|---|---|---|---|---|---|---|---|
| | T | NT | NT | T | T | NT | NT | T |
| Vorhersage korrekt | Nein | Ja | Nein | Nein | Nein | Nein | Nein | Nein |

c) Es ergibt sich eine Vorhersagegüte von 0 %.

| | B1 | B2 | B1 | B2 | B1 | B2 | B1 | B2 |
|---|---|---|---|---|---|---|---|---|
| | T | NT | T | NT | T | NT | T | NT |
| Vorhersage korrekt | Nein | Nein | Nein | Nein | Nein | Nein | Nein | Nein |

d) Zu jedem gemeinsamen Prädiktor gehört eine Menge von zwei oder mehreren Verzweigungen. Der Prädiktor wird sich an das durchschnittliche Verhalten dieser Verzweigungen anpassen. Bei langen Befehlsfolgen wird jedoch die Menge der zu einem Prädiktor gehörenden Verzweigungen nicht konstant sein, sondern sie wird sich im Zeitverlauf verändern. In den Übergangsphasen kann dies zu einer deutlichen Verschlechterung der Vorhersagen führen.

## Aufgabe 4: Cycle-by-Cycle Interleaving (3+5+2+2= 12 Punkte)

Gegeben seien sechs Kontrollfäden (*Threads*) $T_1,...,T_6$, die auf einem mehrfädigen Prozessor ablaufen, der das Cycle-by-Cycle Interleaving Verfahren verwendet. Die Threads haben folgende Eigenschaften:

- $T_1$ gibt zunächst einen Befehl aus, der einen Speicherzugriff erfordert. Danach werden zwei weitere Befehle ausgegeben, die jedoch keine Latenzen bewirken.
- $T_2$ gibt zunächst drei latenzfreie Befehle aus und dann einen Befehl, der auf den Speicher zugreift.
- $T_3$ besteht aus fünf latenzfreien Befehlen.
- $T_4$ gibt zunächst zwei latenzfreie Befehle aus. Der dritte Befehl bewirkt eine Pipeline-Latenz von zwei Taktzyklen. Darauf folgen wieder zwei latenzfreie Befehle.
- $T_5$ hat die gleiche Eigenschaft wie $T_1$.
- $T_6$ besteht aus vier latenzfreien Befehlen.

Die Latenz eines Speicherzugriffs beträgt 10 Taktzyklen.

## Aufgabe 5: Abhängigkeiten ( 8 Punkte)

Es sei folgende Programmsequenz gegeben:

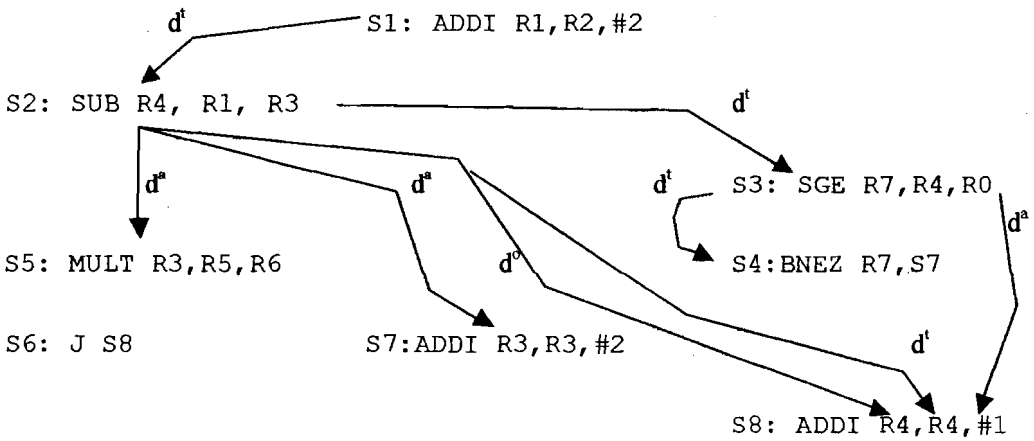
```
S1:  ADDI    R1, R2, #2    ; R1 = R2 + 2
S2:  SUB     R4, R1, R3    ; R4 = R1 - R3
S3:  SGE     R7, R4, R0    ; R4 >= 0 ? Status in R7
S4:  BNEZ    R7, S7        ; wenn ja, gehe zu S7
S5:  MULT    R3, R5, R6    ; R3 = R5 * R6
S6:  J       S8            ; Goto S8
S7:  ADDI    R3, R3, #2    ; R3 = R3 + 2
S8:  ADDI    R4, R4, #1    ; R4 = R4 +1
```

Bestimmen Sie alle Daten- und Steuerflussabhängigkeiten in dieser Programmsequenz. Stellen Sie diese in einem Abhängigkeitsbaum dar.
Kennzeichnen Sie die Pfeile des Baums mit

| | | |
|---|---|---|
| $d^t$ | true dependence | Lesen nach Schreiben |
| $d^o$ | output dependence | Zielregister identisch |
| $d^a$ | anti dependence | Schreiben nach Lesen |

### *Musterlösung:*

Datenabhängigkeiten:




Steuerflussabhängigkeiten: Nur die bedingte Sprunganweisung S4 erzeugt eine Kontrollabhängigkeit. Allerdings gibt es noch eine weitere Steuerflussänderung bei der unbedingten Sprunganweisung S6.

a) Skizzieren Sie das Laufzeitverhalten der Threads bei einfädiger Befehlsverarbeitung.

b) In welcher Reihenfolge werden die Threads auf dem mehrfädigen Prozessor abgearbeitet? Der Prozessor soll mit $T_1$ beginnen und mit jedem Takt zyklisch zwischen den Threads umschalten (*Round-Robin*). Markieren Sie die Latenzzeiten.

c) Vergleichen Sie die Zahl der benötigten Taktzyklen bei ein- und mehrfädiger Verarbeitung.

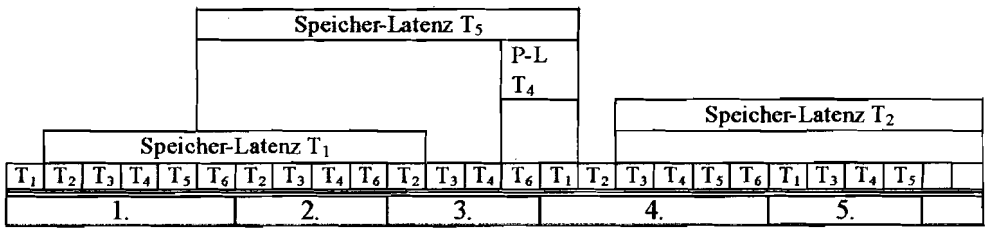
d) Bestimmen Sie den Grad der Prozessoreffizienz für die beiden Fälle aus c). Der Grad der Effizienz ist der Quotient aus Anzahl ausgeführter Befehle und Anzahl Takte der gesamten Ausführung (inkl. Latenz).

## *Musterlösung:*

a) Das Laufzeitverhalten bei einfädiger Befehlsverarbeitung ergibt:
(P-L bedeutet Pipeline-Latenz, x bedeutet Befehlsverarbeitung (*busy*) )

| Thread | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $T_1$ | x | Speicher-Latenz | | | | | | | | | | x | x | |
| $T_2$ | x | x | x | x | Speicher-Latenz | | | | | | | | | |
| $T_3$ | x | x | x | x | x | | | | | | | | | |
| $T_4$ | x | x | x | P-L | | x | x | | | | | | | |
| $T_5$ | x | Speicher-Latenz | | | | | | | | | | x | x | |
| $T_6$ | x | x | x | x | | | | | | | | | | |

b) Ein mehrfädiger Prozessor arbeitet die Threads $T_1$ bis $T_6$ in folgender Weise ab:




c)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| einfädig | 13 | + | 14 | + | 5 | + | 7 | + | 13 | + | 4 | = | 56 Takte |
| mehrfädig | 6 | + | 4 | + | 4 | + | 6 | + | 4 | + | 2 | = | 26 Takte |

d) Zählt man die ‚x' in der Skizze der Aufgabe a), so erhält man 24 ausgeführte Befehle in 56 Takten, die Prozessoreffizienz für den einfädigen Prozessor ist also

$$\frac{24}{56} = 42{,}9\%$$

Für den mehrfädigen Prozessor ergibt sich:

$$\frac{24}{26} = 92{,}3\%$$