Aufgabe 1: Cache-Adressierung

Ihr Cachingsystem soll 32 GiB an Speicher auf Datenwortebene adressieren können. Die Datenwortlänge beträgt 8 Byte, der Cache umfasst 256 KiB und besteht aus 128 Byte großen Blöcken.

a) Betrachten Sie für den Cache die Varianten *Direct Mapped Cache*, *4-Way Set Associative Cache* und *Fully Associative Cache*. Berechnen Sie für jede dieser Varianten die Längen von *Tag*, *Index*, *Offset* und die Adresslänge!

b) Berechnen Sie zu den Adressen 0xBADC0DED und 0x12345432 die Werte für *Tag*, *Index* und *Offset* zu jedem der drei Caches aus Teilaufgabe a)! Geben Sie die Werte jeweils binär und hexadezimal an, wobei führende Nullen weggelassen werden können.

c) Angenommen, zur Realisierung der Ersetzungsstrategie (z.B. LRU, LFU, etc.) werden beim *4-Way Set Associative Cache* 3 Bit pro Block und beim *Fully Associative Cache* 10 Bit pro Block benötigt.
Bei schreibendem Zugriff soll darüber hinaus bei einem *Hit* das Verfahren *Copy-Back* angewendet werden, anderenfalls *Fetch-on-Write*.

Wie viel Speicher muss für die Verwaltung der drei Caches (alle gespeicherten Informationen außer den Nutzdaten) jeweils berücksichtigt werden?

Aufgabe 2: Fully Associative Cache

Auf einem Prozessor mit einer Adresslänge von 8 Bit wird folgende Befehlssequenz ausgeführt (alle Konstanten sind im Hexadezimalsystem angegeben!):

$$
\begin{array}{lrcl}
t_1: & R0 & \leftarrow & \text{0x22} \\
t_2: & R1 & \leftarrow & \text{rsh}(R0) \\
t_3: & memory[+(R0)] & \leftarrow & \text{0x11} \\
t_4: & memory[\text{0x11}] & \leftarrow & R1 \\
t_5: & R2 & \leftarrow & memory[(R1)-] \\
t_6: & memory[R2 + \text{0x22}] & \leftarrow & R2 \\
t_7: & R3 & \leftarrow & memory[R0 - \text{0x12}] \\
t_8: & R0 & \leftarrow & memory[\text{0x23}] \\
t_9: & memory[+(R1)] & \leftarrow & R3
\end{array}
$$

a) Bestimmen Sie die Datenadressen, die sich aus dem Ablauf der Befehlssequenz ergeben, und tragen Sie diese in die unten stehende Tabelle ein.

b) Es wird angenommen, dass der Prozessor einen *Fully Associative Cache*, bestehend aus 2 Blöcken zu je 4 Datenwörtern, besitzt. Die verwendete Ersetzungsstrategie ist *Least Recently Used*. Lese- und Schreibzugriffe sollen dabei gleich behandelt werden, d.h. der betreffende Datenblock wird in jedem Fall in den Cache geladen.
Bestimmen Sie für jede Adresse *Tag* und *Offset*. Tragen Sie in die Tabelle ein, ob es sich beim jeweiligen Speicherzugriff um ein *read(r)* oder *write(w)* handelt sowie ob es ein *hit(h)* oder *miss(m)* ist. Die Spalte *a* (*access*) soll den letzten Zugriff protokollieren, die Spalte *v* soll das Valid-Bit enthalten. Verwenden Sie für *Tag* und *Offset* die binäre, und für die Adresse die hexadezimale Notation!

						Block 0			Block 1		
t	Adresse	Tag	Offset	h/m	r/w	Tag	v	a	Tag	v	a
t_0	0x —	—	—	—	—	000000	0	0	000000	0	0
t_1											
t_2											
t_3											
t_4											
t_5											
t_6											
t_7											
t_8											
t_9											

c) Berechnen Sie die *miss rate* der oben angeführten Speicherzugriffssequenz!

Aufgabe 3: Set Associative Cache

Für einen anfangs leeren Cache ist eine Sequenz von Adresszugriffen gegeben. Bei dem Cache handelt es sich um einen *2-Way Set Associative Cache* mit 4 Sets. Pro Block können 16 Datenwörter gespeichert werden. Die Datenwortlänge beträgt ein Byte, die Adressierung erfolgt auf Datenwortebene. Als Ersetzungsstrategie gilt *Least Frequently Used* (LFU). Falls unter den Ersetzungskandidaten mehrere Blöcke innerhalb eines Sets in Frage kommen, wird der Block mit der niedrigeren Blocknummer ausgewählt. Nach dem Ersetzen eines Blocks startet die zugehörige Anzahl der Zugriffe wieder bei 1.

a) Wie viele Bytes Nutzdaten können in diesem Cache maximal gespeichert werden?

b) Markieren Sie Tag, Index und Offset im folgenden 8-Bit Register, wenn dieses eine Adresse enthalten würde!

	7	6	5	4	3	2	1	0	
MSB									LSB

c) Tragen Sie in der nachfolgenden Tabelle ein, welches Cache-Set die jeweils angegebene Adresse anspricht und ob es sich bei dem Zugriff um einen *Hit* oder einen *Miss* handelt.
Geben Sie bei einem *Hit* an, in welchem Block des Sets die zugehörigen Daten gespeichert sind. Geben Sie bei einem *Miss* an, welcher Block des Sets überschrieben wird und von welchen Adressen Daten in diesen Block geladen werden. Verwenden Sie für diesen Zugriff auf den Hauptspeicher die Notation *mem[x-y]*, wobei x die erste und y die letzte Adresse des Adressbereichs angibt, von dem die Daten in den Cache geladen werden.

Die Adressen sind in dezimaler Form gegeben und sollen zunächst in Binärdarstellung umgewandelt werden. Die erste Zeile ist bereits vorausgefüllt.

Adresse (dezimal)	Adresse (binär)	Hit/Miss	Set	Block	Inhalt	Anzahl Zugriffe
65	0100 0001	Miss	0	0	mem[64-79]	1
77						
111						
222						
42						
121						
110						
48						
163						
208						
242						
220						
78						
120						
51						

d) Wie hoch ist die *Hit-Rate* der oben angeführten Speicherzugriffssequenz?

e) Wie viel Prozent des Caches blieben unbenutzt, wurden also nicht beschrieben?

Aufgabe 4: Direct Mapped Cache

Gegeben ist ein Prozessor mit einer Adresslänge von 12 Bit und einer Datenwortlänge von 8 Bit.

Der integrierte *Direct Mapped Cache* besitzt 4 Blöcke und speichert 4 Byte pro Block. Für den schreibenden Zugriff wird das Verfahren *Copy-Back* verwendet, falls sich der Block bereits im Cache befindet, andernfalls *Fetch-On-Write*.

a) Skizzieren Sie die Struktur des Caches. Berücksichtigen Sie in Ihrer Skizze außerdem, welche Informationen zusätzlich zu den Nutzdaten für die Verwaltung des Caches gespeichert werden müssen. Geben Sie für die Verwaltungsinformationen die genaue Länge in Bit an!

b) Zeigen Sie, wie sich die Verwaltungsdaten des anfangs leeren Caches verändern, wenn der Prozessor in der gegebenen Reihenfolge die folgenden lesenden (r) oder schreibenden (w) Speicherzugriffe ausführt:

r(0x9EA) - r(0x2FC) - r(0x9EB) - w(0x7A2) - r(0x03E) - w(0x03F) - r(0x863) - w(0x9E8) - r(0xA0F)

Geben Sie nach jedem Speicherzugriff die aktuell gespeicherten Werte der Verwaltungsdaten im gesamten Cache an. Geben Sie außerdem zu jedem Speicherzugriff an, ob er ein *Hit* oder *Miss* ist!

Aufgabe 5: Multilevel Caching

Gehen Sie von einem Prozessor mit folgenden Eckdaten aus:

- Prozessortakt: 3 GHz
- Zugriffsdauer auf L1-Cache: 3 Taktzyklen
- Zugriffsdauer auf L2-Cache: 45 Taktzyklen
- Zugriffsdauer auf Hauptspeicher: 150 Taktzyklen

Hinweis: In der Zugriffsdauer auf den L2-Cache ist die Zugriffsdauer für den *Miss* auf den L1-Cache bereits inkludiert. Selbiges gilt für die Zugriffsdauer auf den Hauptspeicher.

a) Skizzieren Sie das System mit CPU L1-Cache, L2-Cache und Hauptspeicher.

b) Angenommen, die CPU führt 100.000 Adressanfragen aus. Die *Hit-Rate* (h) für den L1-Cache betrage 90% und für den L2-Cache 96%. Wie viele *Misses* treten beim L1-Cache bzw. beim L2-Cache auf?

c) Geben Sie die Zugriffsdauer auf den L1-Cache (t_{L1}), auf den L2-Cache (t_{L2}) und auf den Hauptspeicher (t_{main}) in Nanosekunden an!

d) Um welchen Faktor verbessert sich die effektive Speicherzugriffszeit durch Verwendung des L2-Caches? Berechnen Sie dazu $t_{\mathrm{eff\_L1}}$ für das Szenario ohne L2-Cache und $t_{\mathrm{eff\_L2}}$ für das Szenario mit L2-Cache. Der Beschleunigungsfaktor ergibt sich dann als Quotient $t_{\mathrm{eff\_L1}}/t_{\mathrm{eff\_L2}}$.

Hinweis: Die effektive Speicherzugriffszeit wird bei Verwendung eines Caches grundsätzlich nach folgender Formel berechnet:

$$t_{\mathrm{eff}} = h \times t_{\mathrm{cache}} + (1 - h) \times t_{\mathrm{main}}$$

Aufgabe 6: Paging / System-Analyse

Ein System arbeitet mit 32 KiB großen Pages und Frames, die kleinste adressierbare Einheit ist 1 Byte. Zu Beginn werden die folgenden (virtuellen) Adresszugriffe abgearbeitet, wobei Frames in aufsteigender Reihenfolge (bei 0 beginnend) beschrieben werden und als Ersetzungsstrategie *First In – First Out (FIFO)* verwendet wird:

Zeitpunkt	Adresszugriff
t_1	0x150DA
t_2	0x396E1
t_3	0x0A4F7
t_4	0x2688A
t_5	0x12345

Die resultierende Page-Table lautet (ergänzen Sie noch die Zeitpunkte):

Page-Nr	Frame-Nr	Present-Bit	Zeitpunkt
000	00	0	
001	10	1	
010	00	1	
011	00	0	
100	11	1	
101	00	0	
110	00	0	
111	01	1	

a) Wie lang sind in diesem System virtuelle Adressen, wie lang physische Adressen?

b) Wie viel Byte physischen/virtuellen Speichers sind im System adressierbar?

c) Wie lautet die entsprechende physische Adresse zur (virtuellen) Adresse 0x26A3B?

d) Wie verändert sich die Page-Table durch die nachfolgenden beiden Adresszugriffe?

Zeitpunkt	Adresszugriff
t_6	0x35232
t_7	0x02ABC

Hinweis: Sie brauchen nur jene Zeilen eintragen, die sich gegenüber der obigen Darstellung der Page-Table verändern.

Page-Nr	Frame-Nr	Present-Bit	Zeitpunkt
000			
001			
010			
011			
100			
101			
110			
111			

Aufgabe 7: Paging / System-Entwurf

Entwerfen Sie ein System, das mit 15 Bit langen virtuellen Adressen arbeitet. Die Adressierung soll auf Datenwortebene erfolgen, wobei ein Datenwort 1 Byte lang ist. Im System sollen 16 KiB Speicher physisch verbaut sein und die virtuelle Speicherverwaltung soll mit 4 KiB großen Page-Frames arbeiten.

a) Geben Sie den Aufbau der virtuellen und physischen Adressen an und skizzieren Sie die Page-Table. Berücksichtigen Sie dazu auch die unten angegebenen Informationen zur Ersetzungsstrategie!

b) Geben Sie die Zuordnung des physischen Adressbereichs zu den einzelnen Frames an!

c) Der physische Speicherbereich von Frame 0, beginnend bei Adresse 00...0, soll dauerhaft für das Betriebssystem reserviert werden, die weiteren Frames werden initial nach aufsteigender Frame-Nummer vergeben. Als Ersetzungsstrategie soll *Least Frequently Used (LFU)* verwendet werden, wobei 3 Bit für die Darstellung der bisherigen Anzahl an Zugriffen verwendet werden. Falls unter den Ersetzungskandidaten mehrere Frames in Frage kommen, wird der mit der niedrigeren Frame-Nummer ausgewählt. Sie starten auf dem System einen Prozess P, der wie folgt auf den (virtuellen) Speicher zugreift:

0x4CAD – 0x178A – 0x2431 – 0x2B0B – 0x4000 – 0x7DEA – 0x6BAC – 0x4FB1

Zeigen Sie, wie der Prozess P den Speicher nutzt, wenn diese Speicherzugriffssequenz ausgeführt wird! Nehmen Sie an, dass die Page-Table zu Beginn leer ist. Geben Sie bei jedem Speicherzugriff an, ob es sich um einen Page-Fault handelt oder nicht.

Aufgabe 8: Paging / Parallelität

Sie arbeiten mit einem Betriebssystem, das 12 Bit virtuelle Adressen verwendet. 9 Bit davon gehören zum Offset. Die physischen Adressen sind 11 Bit lang. Ersetzungsstrategie ist *Least Recently Used (LRU)*.

Die Programme P1 und P2 laufen parallel zueinander, wobei jedes Programm eine eigene Page-Table besitzt. Die Programme werden wie folgt ausgeführt (acht Takte):

P1 – P2 – P2 – P1 – P2 – P2 – P1 – P1

Die Programme greifen zu ihrem jeweiligen Abarbeitungsschritt auf folgende virtuelle Adressen zu:

Schritt	P1	P2
1	0x800	0xB53
2	0xB0B	0x71F
3	0xBCD	0xA45
4	0x932	0x0FF

a) Berechnen Sie die Größe des virtuellen und physischen Speichers sowie die Framegröße, wenn auf Byte-Ebene adressiert wird.

b) Tragen Sie in nachfolgender Tabelle ein, welche Page in welchem Frame liegt und zu welchem Programm sie gehört. Tritt ein Page-Fault auf, aktualisieren Sie den Inhalt des entsprechenden Frames. Liegt die Page bereits im physischen Speicher, tragen Sie beim entsprechenden Frame 'H' ein.
Die ersten beiden Zeilen sind bereits vorausgefüllt.

	Schritt		Frame-Nummer			
Takt	P1	P2	00	01	10	11
1	1		P1: 0x4			
2		1		P2: 0x5		
3						
4						
5						
6						
7						
8						

c) Geben Sie die Page-Tables von P1 und P2 nach Abarbeitung aller Schritte an. Befüllen Sie dabei nur jene Zeilen, die während des Programmablaufs modifiziert wurden!

Page-Table von Programm P1:

PageNr	Frame	Present-Bit
0x0		
0x1		
0x2		
0x3		
0x4		
0x5		
0x6		
0x7		

Page-Table von Programm P2:

PageNr	Frame	Present-Bit
0x0		
0x1		
0x2		
0x3		
0x4		
0x5		
0x6		
0x7		