Universität Paderborn
Institut für Elektrotechnik und Informationstechnik
Fachgebiet *Datentechnik*
Prof. Sybille Hellebrand

Klausur

Grundlagen der Rechnerarchitektur

13. Februar 2018

Punkteverteilung								
Aufgabe	1	2	3	4	5	Σ	BP	Σ
maximale Punkte	10	20	20	20	20	90	9	99
erreichte Punkte								

Note:	

Aufkleber

Name:	
Matrikelnummer:	
Studienrichtung:	

Hinweise:
Für die Lösung der Klausuraufgaben sind ausschließlich die Aufgabenblätter zu verwenden. Lösungsangaben außerhalb der Aufgabenblätter („Schmierzettel“, etc.) werden bei der Bewertung nicht berücksichtigt!

Beschriften Sie jede Doppelseite mit Ihrer Matrikelnummer!

Mit Bleistift oder der Korrekturfarbe rot angefertigte Lösungen werden nicht bewertet!

Die Verwendung von „Tipp-Ex“ oder „Tintenkiller“ ist untersagt.

Es ist ein handgeschriebener DIN-A4 Zettel als Hilfsmittel zugelassen!

Es sind keine weiteren Hilfsmittel zugelassen!

Aufgabe 1: (Leistungsbewertung) 10 Punkte

a) Für einen Prozessor soll der mögliche Speedup berechnet werden, der durch den Einsatz einer Fließkommaeinheit erreicht werden kann. Das verwendete Benchmark-Programm hat einen Anteil an Fließkommaoperationen von 30%. Wie hoch ist der maximale Speedup für dieses Programm, wenn mit Hilfe der Fließkommaeinheit die Fließkommaoperationen 3 mal schneller ausgeführt werden können?

Bitte kreuzen Sie an. (3 Punkte)

- ☐ $\frac{5}{4}$
- ☐ $\frac{7}{6}$
- ☐ $\frac{6}{5}$
- ☐ Keine Lösung ist richtig.

b) Ein Prozessorhersteller testet mit Hilfe eines Benchmark-Programms die Leistungsfähigkeit seines Prozessors. Das Programm führt hierzu 10^7 Instruktionen aus. Der Prozessor hat einen idealen CPI-Wert von 2 und wird mit einer Taktrate von 2 GHz betrieben. Der Fehlgriffsaufwand (*engl.*: miss-penalty) des verwendeten Caches liegt bei 50 CPU-Taktzyklen. In dem verwendeten Benchmark-Programm tritt bei Befehlen eine Fehlgriffsrate (*engl.*: miss-rate) von 2% auf und bei Daten von 4%, wobei die Häufigkeit der Datenzugriffe bei 50% liegt. Wie hoch ist die Ausführungszeit des Benchmark-Programms auf der beschriebenen Architektur?

Bitte kreuzen Sie an! (4 Punkte)

- ☐ 11 ms
- ☐ 20 ms
- ☐ 25 ms
- ☐ Keine Lösung ist richtig.

c) Der Prozessorhersteller möchte nun die Speicherzugriffszeit verringern. Hierzu wird ein Level-2-Cache eingebaut. Die technischen Daten des Level-1-Caches und des Level-2-Caches sind wie folgt:

Bezeichnung		**Level-1-Cache**	**Level-2-Cache**
Trefferzeit (*engl.:* hit-time)		2 Takte	9 Takte
Fehlgriffsaufwand (*engl.:* miss-penalty)		mittlere Zugriffszeit L2	50 Takte
Fehgriffsrate (*engl.:* miss-rate)	Befehle	2%	2%
	Daten	4%	2%

Die Wahrscheinlichkeit für Datenzugriffe beträgt 50%.

Wie viele Takte werden im Durchschnitt benötigt, um auf den Speicher zuzugreifen?

Bitte kreuzen Sie an. (3 Punkte)

- ☐ 13 Takte
- ☐ 50 Takte
- ☐ 15 Takte
- ☐ Keine Lösung ist richtig.

Aufgabe 2: (Assembler) 20 Punkte

Die Tribonaccifolge ist rekursiv definiert durch:

$$T(n) = T(n-1) + T(n-2) + T(n-3) \quad \text{mit} \quad T(0) = T(1) = T(2) = 1$$

Gegeben ist die folgende Prozedur `trib` in MIPS-Assembler, welche die n-te Tribonaccizahl, für $n \geq 0$, *iterativ* berechnet. Entsprechend der MIPS-Konventionen für die Registervergabe befindet sich bei Aufruf der Prozedur der Eingabewert n im Register `$a0` und der Rückgabewert $T(n)$ wird im Register `$v0` erwartet.

```
trib:   slti    $t0,    $a0,     3      # wenn n<3 dann gib 1 zurueck
        beq     $t0,    $zero,   L1     #
        add     $v0,    $zero,   1      #
        jr      $ra                     #

L1:                                     # wenn n>=3 dann iterative Berechnung
        add     $t0,    $zero,  1       # initialisiere $t0 mit T(0)
        add     $t1,    $zero,  1       # initialisiere $t1 mit T(1)
        add     $t2,    $zero,  1       # initialisiere $t2 mit T(2)
        subi    $t3,    $a0,    2       # initialisiere $t3 als Schleifenzaehler

L2:     add     $t4,    $t0,    $t1     # bestimme T(n)
        add     $t4,    $t4,    $t2     #

        add     $t0,    $zero,  $t1     # $t0 haelt neues T(n-3)
        add     $t1,    $zero,  $t2     # $t1 haelt neues T(n-2)
        add     $t2,    $zero,  $t4     # $t2 haelt neues T(n-1)

        addi    $t3,    $t3,    -1      # erniedrige Schleifenzaehler
        bne     $t3,    $zero,  L2      # Abbruch wenn Schleifenzaehler gleich null

        add     $v0,    $zero,  $t4     # gib T(n) zurueck
        jr      $ra                     #
```

a) Bestimmen Sie $I_c(\texttt{trib}, n)$, die Anzahl der Instruktionen, die in der Prozedur `trib` in Abhängigkeit von n ausgeführt werden, wenn sie mit $n \geq 3$ aufgerufen wird.

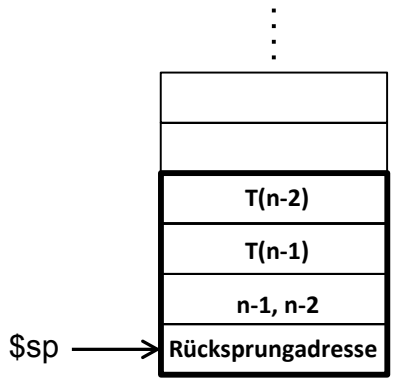
(4 Punkte)

$I_c(\texttt{trib}, n) =$ _____

b) Bestimmen Sie $I_c(\texttt{trib}, 3)$, die Anzahl der Instruktionen, die in der Prozedur `trib` für $n = 3$ ausgeführt werden. (1 Punkte)

$I_c(\texttt{trib}, 3) =$ _____

c) Vervollständigen Sie die untenstehende Assemblerprozedur `tribR`, welche die Berechnung von $T(n)$ *rekursiv* implementiert. Der Prozeduraufrufrahmen ist vier Worte groß und folgendermaßen aufgebaut: (10 Punkte)




```
tribR:  slti    $t0,    $a0,    3       # wenn n<3 dann gib 1 zurueck
        beq     $t0,    $zero,  L1
        add     $v0,    $zero,  1
        jr      $ra
                                        # wenn n>=3 dann rekursive Berechnung
L1:     ____________________________    # schaffe Platz am Stack
        ____________________________    # sichere Ruecksprungadresse

        addi    $a0,    $a0,    -1      # bestimme Argument n-1
        sw      $a0,    __________      # sichere Argument n-1 auf den Stack
        ____________________________    # berechne T(n-1)
        sw      $v0,    __________      # sichere T(n-1) auf den Stack

        lw      $a0,    __________      # lies n-1 vom Stack
        addi    $a0,    $a0,    -1      # bestimme Argument n-2
        sw      $a0,    __________      # sichere n-2 auf den Stack
        ____________________________    # berechne T(n-2)
        sw      $v0,    __________      # sichere T(n-2) auf den Stack

        lw      $a0,    __________      # lies n-2 vom Stack
        addi    $a0,    $a0,    -1      # bestimme Argument n-3
        ____________________________    # bestimme T(n-3)

        lw      $t0,    __________      # lies T(n-1) vom Stack
        lw      $t1,    __________      # lies T(n-2) vom Stack

        add     $t3,    $t0,    $t1     # bestimme T(n) =
        add     $v0,    $v0,    $t3     # T(n-1)+T(n-2)+T(n-3)

        ____________________________    # stelle Ruecksprungadresse wieder her
        ____________________________    # gib Platz am Stack wieder frei
        jr      $ra
```

d) Bestimmen Sie $I_c(\texttt{tribR}, 3)$ und vergleichen Sie die Performance der zwei Prozeduren unter der Annahme einer MIPS-Einzyklenimplementierung mit der Taktfrequenz von $f = 200\,\text{MHz}$. Um welchen Faktor ist die iterative Implementierung für $n = 3$ schneller? (5 Punkte)

$I_c(\texttt{tribR}, 3) =$ _____

Die iterative Implementierung ist um den Faktor _____ schneller.

Aufgabe 3: (Datenpfad) 20 Punkte

Der MIPS-Instruktionssatz, soll um den Befehl `swap` erweitert werden. Dem Befehl werden zwei Registeradressen übergeben (`r1` und `r2`). Nach Ausführung des Befehls sollen die Registerinhalte ausgetauscht worden sein, d.h. wenn die Registerinhalte vor dem Swap-Befehl `R[r1] = x` und `R[r2] = y` sind, dann muss nach dem Swap-Befehl `R[r1] = y` und `R[r2] = x` gelten. Es dürfen ausschließlich die beiden angegebenen Register verwendet werden.

Hinweis: Mit Hilfe des XOR swap Algorithmus können zwei Zahlen getauscht werden. Algorithmus 1 zeigt den Pseudocode des Algorithmus.

Algorithm 1 XOR `swap` Algorithmus

1: $x \leftarrow x$ **xor** y
2: $y \leftarrow y$ **xor** x
3: $x \leftarrow x$ **xor** y

a) Abbildung 1 zeigt eine Einzyklenimplementierung des MIPS Instruktionssatzes. Ist die Ausführung des zuvor definierten Befehls `swap` mit Hilfe des gezeigten Datenpfades möglich? Begründen Sie Ihre Antwort kurz. (3 Punkte)

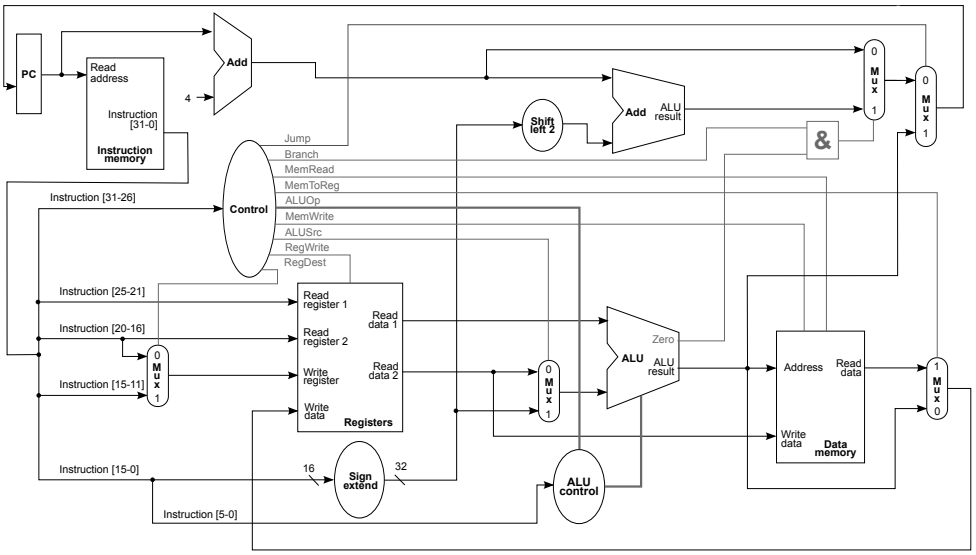


Abbildung 1: Einzyklenimplementierung des MIPS Instruktionssatzes

Antwort: .....

.....

.....

b) Abbildung 2 zeigt den Zustandsautomaten, der die Steuerung einer Mehrzyklenimplementierung des MIPS-Instruktionssatzes beschreibt. Erweitern Sie den Automaten so, dass der neue Befehl `swap` unterstützt wird. Beachten Sie ggf. benötigte Änderungen des Datenpfads. Ersatz in Abbildung 4. (10 Punkte)

Hinweis: Bei `ALUOp=10` entscheidet der ALU-Kontroller welche Operation von der ALU ausgeführt wird.

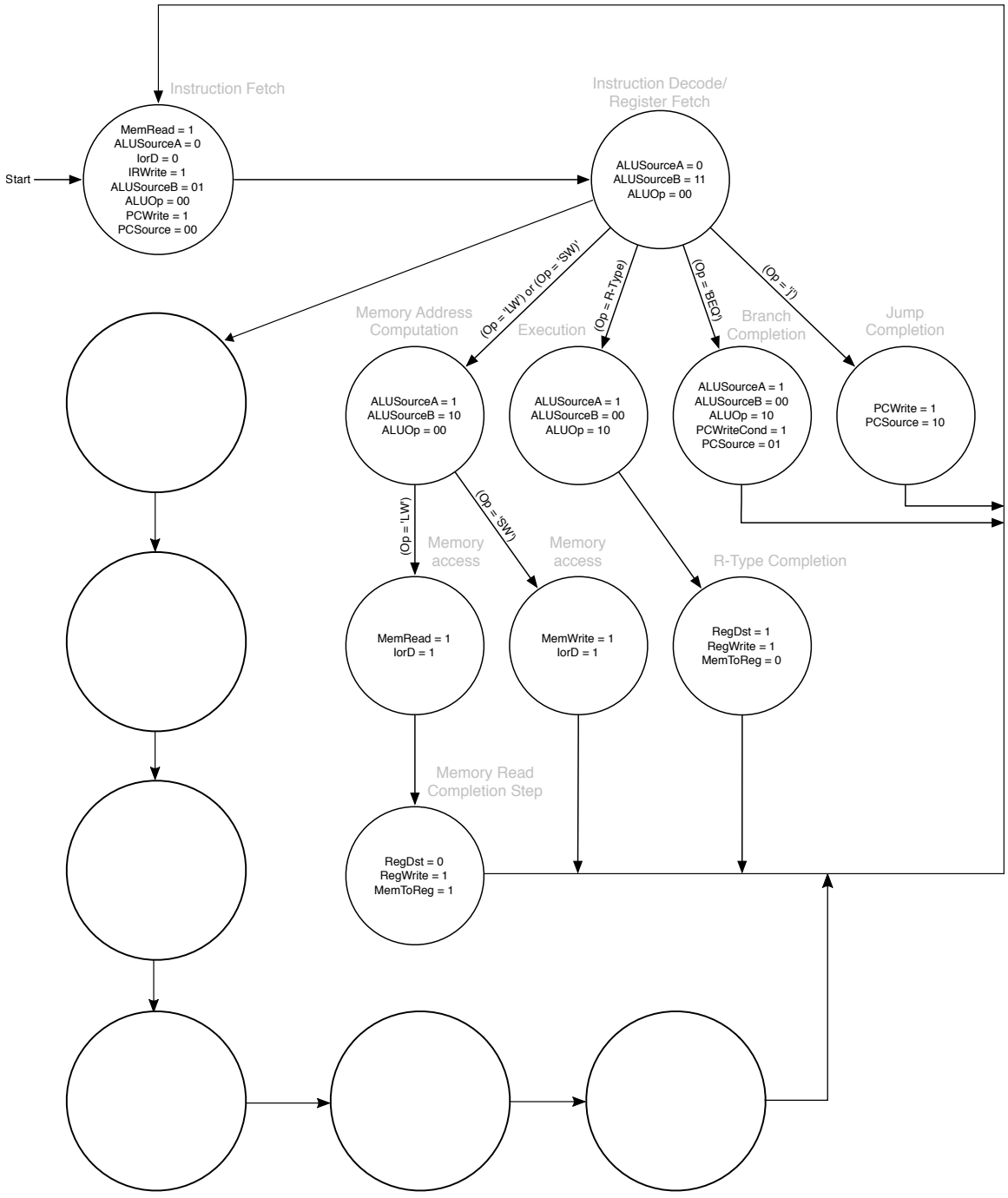


Abbildung 2: Zustandsautomat der Steuerung der Mehrzyklenimplementierung

c) Erweitern Sie den Datenpfad in Abbildung 3 so, dass der Befehl `swap` unterstützt wird. Zeichnen Sie hierzu alle notwendigen Daten- und Kontrollleitungen in die Abbildung 3 ein. Ersatz in Abbildung 5. (3 Punkte)

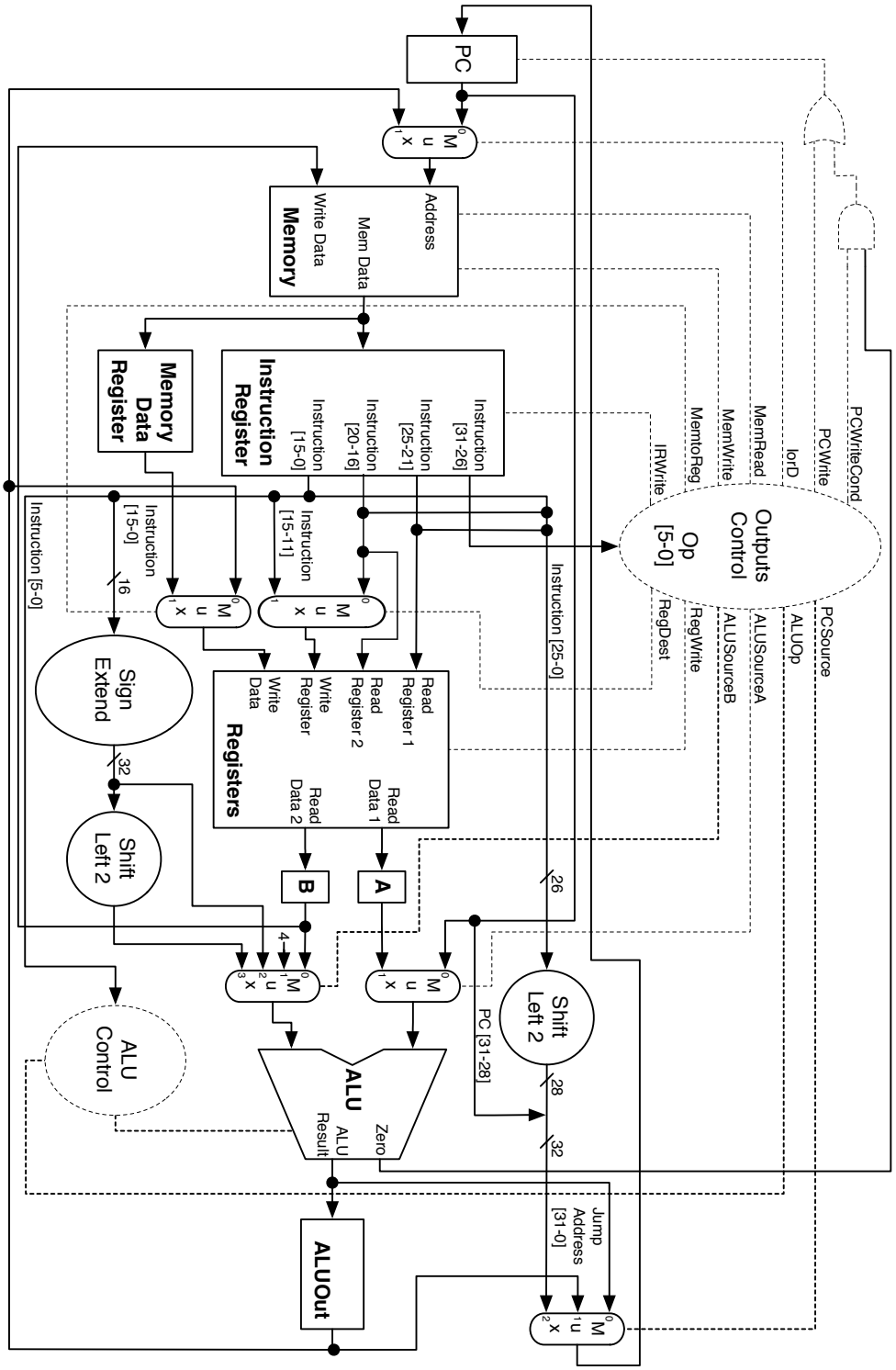


Abbildung 3: Mehrzyklenimplementierung des MIPS Instruktionssatzes

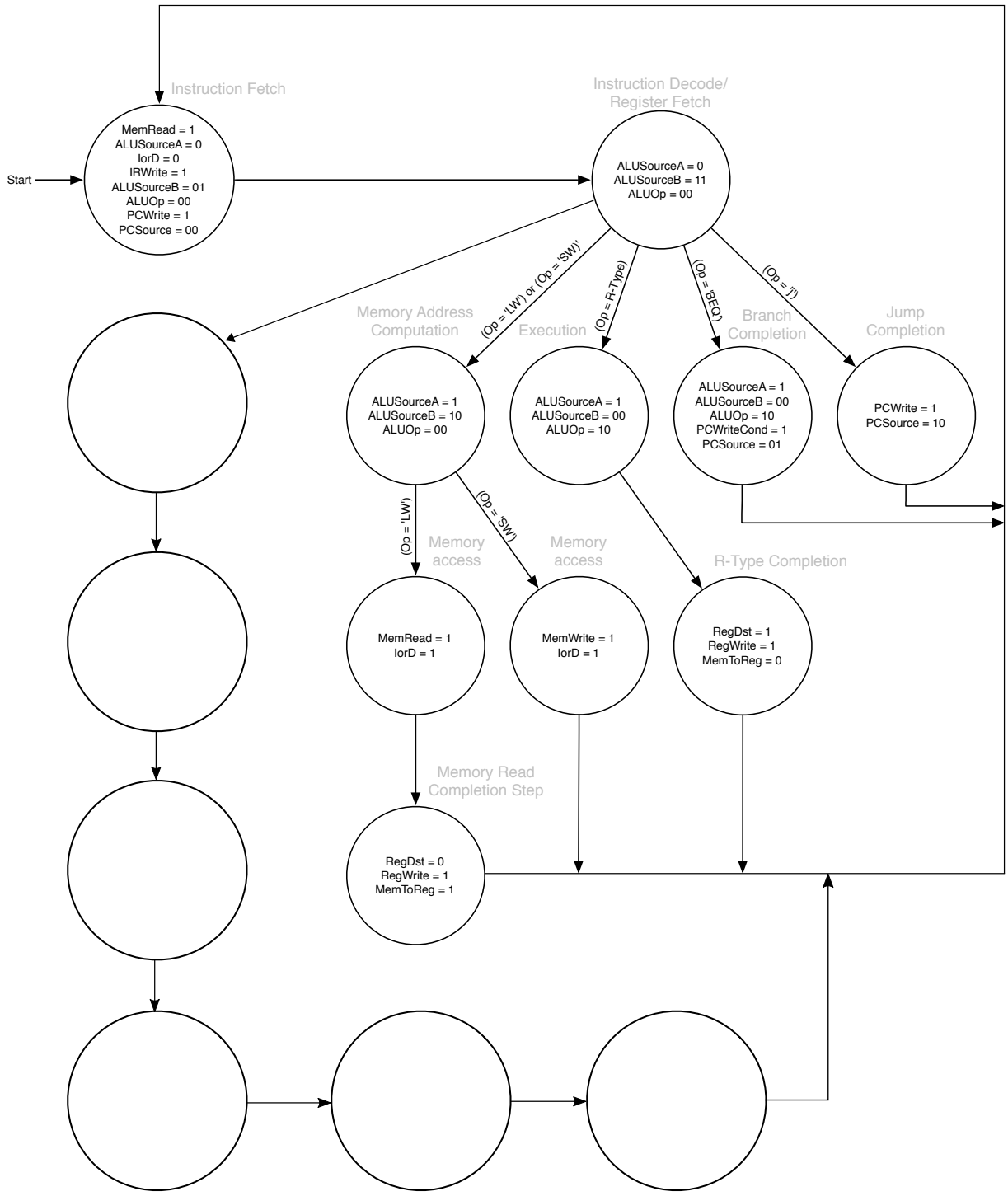


Abbildung 4: **Ersatz für b), ungültige Lösung streichen!**

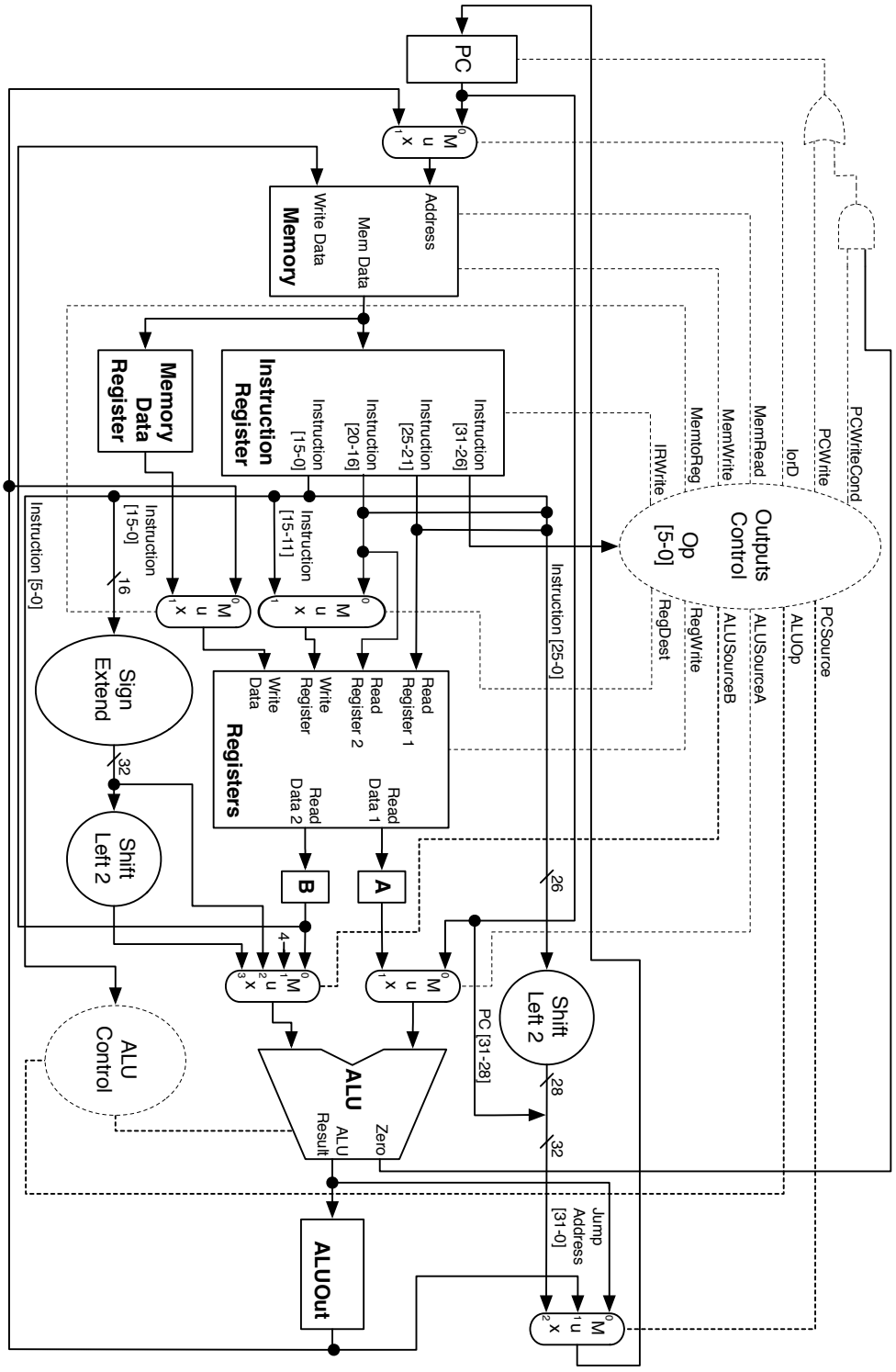


Abbildung 5: **Ersatz für c), ungültige Lösung streichen!**

d) Geben Sie das Befehlsformat für den Befehl `swap` an. Wählen Sie hierzu einen der drei aus der Vorlesung bekannten Befehlstypen. Tragen Sie sowohl den Befehlstypen, die Bezeichnungen der einzelnen Befehlsbereiche, das Start- und Endbit der einzelnen Bereiche und den Inhalt der einzelnen Bereiche in die Abbildung 6 ein. (4 Punkte)

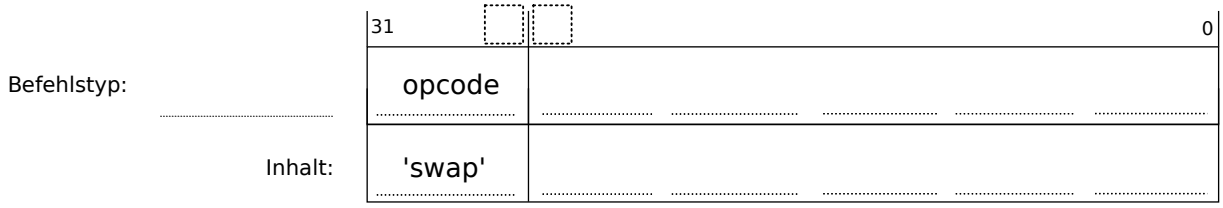


Abbildung 6: Befehlsformat für den Befehl `swap`

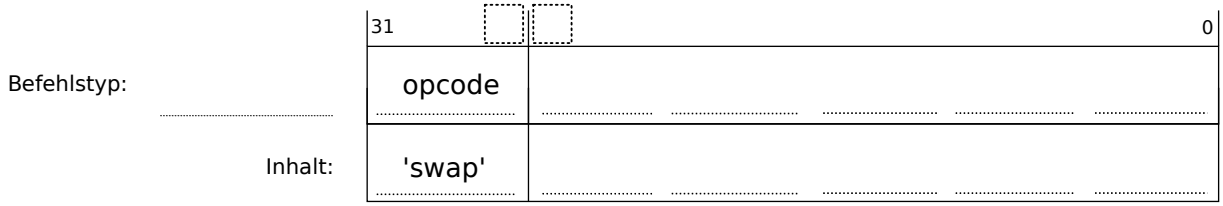


Abbildung 7: **Ersatz, ungültige Lösung streichen!**

Aufgabe 4: (Speicher) 20 Punkte

Gegeben sei ein Computer, der zur Beschleunigung der Speicherzugriffe auf den byteadressierten Arbeitsspeicher mit einem Adressraum von 4 GB (32-Bit-Adressen) einen Cache mit 8 Rahmen vorsieht, wobei jeder Rahmen 4 Worte zu je 32 Bit umfasst. Der Cache sei als 2-fach mengenassoziativer Cache (2-way-set-associative) organisiert.

a) Bestimmen Sie die Größen von Tag, Index, Wortoffset und Byteoffset und zeichnen Sie die Grenzen durch senkrechte Striche in Abbildung 8 ein. Beschriften Sie die entstehenden Felder mit der jeweiligen Bitanzahl und Bezeichnung.
Hinweis: Die Felder müssen *nicht* in den richtigen Proportionen gezeichnet werden. (2 Punkte)

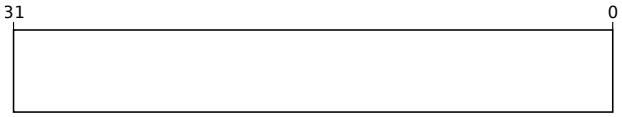


Abbildung 8: Adressaufteilung

b) Wie viel Speicher wird für die Implementierung eines solchen Caches (inkl. Overhead) benötigt? Geben Sie den Rechenweg und das Ergebnis in Byte an.
Hinweis: Für den Zustand eines Cache-Rahmens wird genau ein Statusbit verwendet (Valid-Bit). (3 Punkte)

Cache-Größe (in Byte): _____

c) Der Prozessor lädt nun sequentiell die Daten der folgenden Byteadressen (führende Nullen werden nicht mit angegeben):

```
t=1:   0xc25  = 1100 0010 0101
t=2:   0x6db  = 0110 1101 1011
t=3:   0x768  = 0111 0110 1000
t=4:   0x6d0  = 0110 1101 0000
t=5:   0xa5a  = 1010 0101 1010
t=6:   0x0a1  = 0000 1010 0001
t=7:   0xaf3  = 1010 1111 0011
t=8:   0x6d7  = 0110 1101 0111
t=9:   0x690  = 0110 1001 0000
```

Initial ist der Cache leer. Geben Sie nach jedem Speicherzugriff den Zustand des Caches an. Füllen Sie hierzu die nachfolgenden Tabellen aus. Markieren Sie alle Hits. Geben Sie an, welche Ersetzungsstrategie Sie verwenden und begründen Sie ihre Wahl kurz.

Hinweis: Schreiben Sie z.B. *a* - *b* für den Datenbereich von Adresse *a* bis *b*.

(15 Punkte)

Ersetzungsstrategie: _____

Begündung: .....

.....

.....

.....

t=1: 0xc25

Index	**Tag**	**Datenbereich**	**Hit?**
2	110000	0xc20-0xc2f	☐ Ja

t=2: 0x6db

Index	**Tag**	**Datenbereich**	**Hit?**
			☐ Ja

t=3: 0x768

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=4: 0x6d0

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=5: 0xa5a

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=6: 0x0a1

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=7: 0xaf3

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=8: 0x6d7

Index	Tag	Datenbereich	Hit?
			☐ Ja

t=9: 0x690

Index	Tag	Datenbereich	Hit?
			☐ Ja

Ersatztabellen, **ungültige Tabellen streichen!**:

t=____:

Index	**Tag**	**Datenbereich**

Hit?

☐ Ja

t=____:

Index	**Tag**	**Datenbereich**

Hit?

☐ Ja

t=____:

Index	**Tag**	**Datenbereich**

Hit?

☐ Ja

t=____:

Index	**Tag**	**Datenbereich**

Hit?

☐ Ja

t=____:

Index	**Tag**	**Datenbereich**

Hit?

☐ Ja

Aufgabe 5: (Pipelining) 20 Punkte

Wenn ein Programm auf einem Prozessor mit einer 5-stufigen MIPS-Pipeline ausgeführt wird, können verschiedene Hazards zu einem Pipeline Stall führen. Gegeben sei für diese Aufgabe folgendes Programm, welches zu Data Hazards führt:

```
add $t1, $t2, $t3
lw  $t4, 0($t1)
sub $t5, $t4, $t3
or  $t7, $t1, $t6
```

a) Wie heißen die drei Hazardsorten die beim Pipelining auftreten können? Ordnen Sie sie ihrer jeweiligen Beschreibung zu. (3 Punkte)

i) _____-Hazard:
Die HW unterstützt die gegebene Ausführungsreihenfolge von Operationen nicht.

ii) _____-Hazard:
Der Operand eines Befehls ist das Ergebnis einer anderen Operation, die noch in einer nachfolgenden Stufe der Pipeline bearbeitet wird.

iii) _____-Hazard:
Bedingte Sprunganweisungen führen zum Anhalten der Pipeline, solange bis das Sprungziel berechnet und die Sprungbedingung ausgewertet ist.

b) Betrachten Sie zuerst wie die Befehle in einer Pipeline ohne Optimierungen (also nur mit Pipeline Stall) abgearbeitet werden, indem Sie dafür Abbildung 9 ausfüllen. Beachten Sie, dass Register im selben Takt gelesen und beschrieben werden können und in diesem Fall den neuen Wert zurückgeben. Ersatztabelle in Abbildung 12.

(5 Punkte)

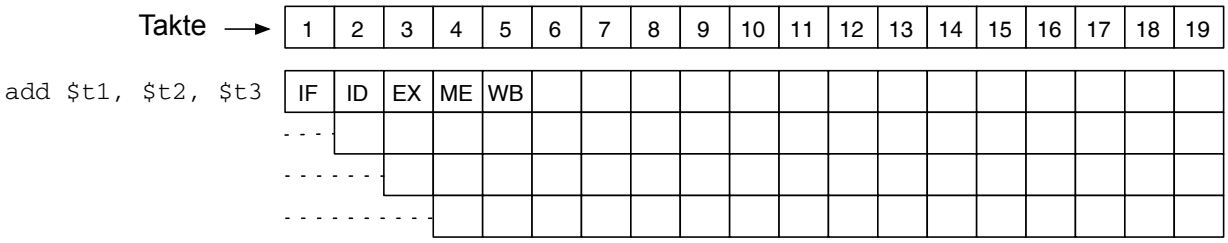


Abbildung 9: Pipelinebelegung mit Pipeline Stall.

```
add $t1, $t2, $t3
lw  $t4, 0($t1)
sub $t5, $t4, $t3
or  $t7, $t1, $t6
```

c) Füllen Sie nun Abbildung 10 aus für eine Pipeline, die zusätzlich Forwarding als Optimierung bietet. Verdeutlichen Sie alle Stellen, an denen Forwarding eingesetzt wird, mit Pfeilen. Ersatztabelle in Abbildung 13. (5 Punkte)

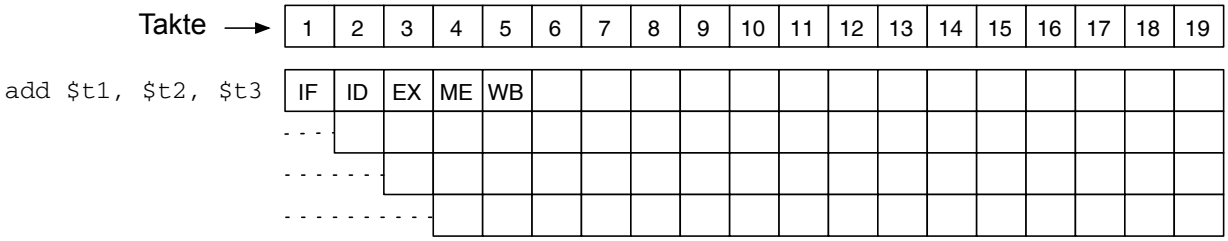


Abbildung 10: Pipelinebelegung mit Forwarding.

d) Wie heißt der spezielle Konflikt, der in **c)** noch auftritt? (2 Punkte)

---

e) In der Vorlesung haben Sie eine weitere Möglichkeit kennengelernt mit der Sie Data Hazards auflösen können. Wenden Sie diese Methode nun hier zusätzlich zum Forwarding an, um alle Data Hazards in dem Programm aufzulösen. Füllen Sie dafür Abbildung 11 aus. Verdeutlichen Sie wiederum alle Stellen, an denen Forwarding eingesetzt wird, mit Pfeilen. Ersatztabelle in Abbildung 14. (5 Punkte)

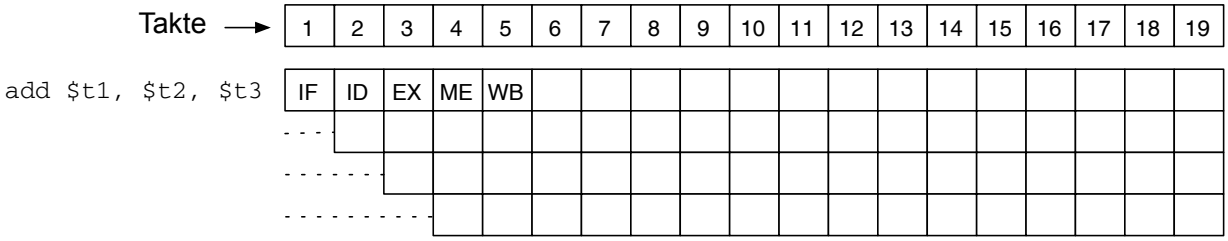


Abbildung 11: Pipelinebelegung mit allen Optimierungen.

Ersatztabellen für **b) - e)**.

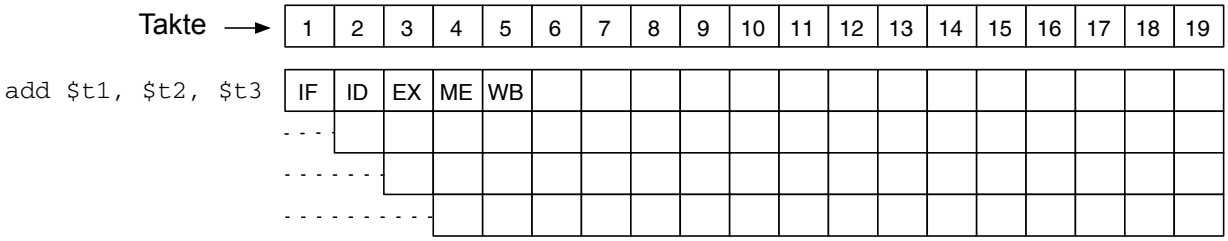


Abbildung 12: Ersatztabelle für Pipelinebelegung mit Pipeline Stall.
Ungültige Lösungen streichen!

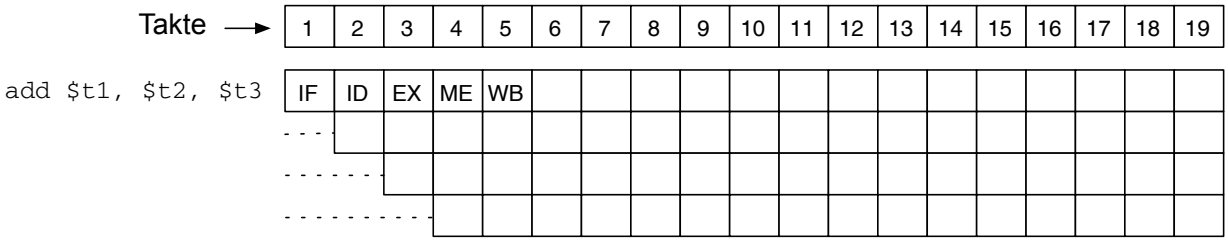


Abbildung 13: Ersatztabelle für Pipelinebelegung mit Forwarding.
Ungültige Lösungen streichen!

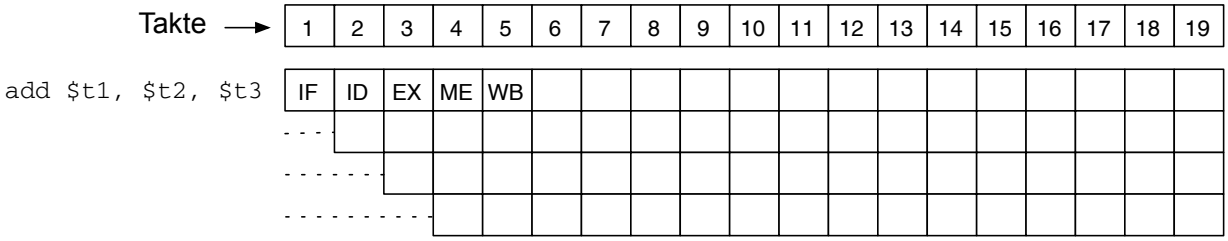


Abbildung 14: Ersatztabelle für Pipelinebelegung mit allen Optimierungen.
Ungültige Lösungen streichen!

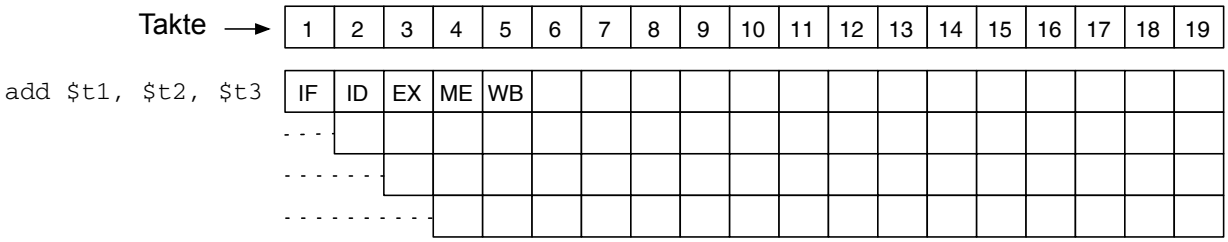


Abbildung 15: Generische Ersatztabelle. Unbedingt erklären für welchen Aufgabenteil sie gelten soll. **Ungültige Lösungen streichen!**