Rechnerarchitektur Übung 6 Aufgabe 2

Erweiterter Prozessor (siehe Abbildung 1):

addInMem:

`addInMem $r0, <offset>($r1)` `<offset>($r1) = <offset>($r1) + $r0`

Für diesen Befehl wird in der ALU die referenzierte Speicheradresse berechnet, ähnlich wie bei Store und Load Word.
Dazu muss ALUSrc auf 1 gesetzt werden. Dadurch wird in ReadData der Wert an dieser Adresse verfügbar. Dieser wird in einen Addierer gegeben (ALUControl: ADD), gemeinsam mit RD2, also dem Wert des zweiten Registers. Dieser Ergebnis wird dann als WD in den Data Memory geführt. Damit allerdings die Store Word Funktion dadurch nicht zerstört wird muss dies durch einen Multiplexer geschehen, der nur bei diesem Befehl das Ergebnis des Addierers in den Data Memory als den zu schreibenden Wert leitet. Für diesen Multiplexer wurde ein neues Signal zur Control Unit hinzugefügt: MemWriteSrc. Für addInMem muss dieses den Wert 1 annehmen, ansonsten bleibt es 0. Damit der Wert auch tatsächlich geschrieben wird, muss auch das MemWrite Signal auf 1 gesetzt werden. Das Ergebnis wird also wieder direkt in den Data Memory geschrieben und nicht in ein Register, daher muss RegWrite auf 0 gesetzt werden.

Veränderung der Taktlänge:
Dieser Befehl ist mit Store Word bezüglich der Taktlänge vergleichbar. Es kommt nur zusätzlich ein Addierer (125ps) und ein 2:1 Multiplexer hinzu (30ps).

Ohne diesen Befehl (Äquivalente Befehlsabfolge)

```
lw $t0, <offset>($r1)
add $t0, $t0, $r0
sw $t0, <offset>($r1)
```

Es wird also statt bisher drei nur mehr ein Taktzyklus benötigt, also zwei Taktzyklen gespart.

addSub:

```
addSub $r0, $r1, $r2, $r3                    $r0 = $r1 + $r2 - $r3
```

Für diesen Befehl brauchen wir ein 4. Register ($r3). Dazu wird der Shift Amount des R Typs verwendet, der sich an den Bits 10:6 befindet. Dies wird als A4 ins Register File hinzugefügt und der Wert dieses Register dann als RD4 verfügbar. Register $r1 und $r2 werden in der ALU miteinander addiert (ALUSrc: 0, ALUControl: ADD). Dieses Zwischenergebnis wird daraufhin in einen Subtrahierer gegeben, gemeinsam mit RD4, dem Wert des Registers $r3. Das Ergebnis der Subtraktion soll im nächsten Schritt in das Register $r0 geschrieben werden. Dafür muss es in den WD3 Eingang des Register Files gelangen. Damit bisherige Funktionalität behalten bleibt wurde dafür der Multiplexer, der auswählt welches Signal nach WD3 gelangt, von 2 auf 4 Bit erweitert, damit wir hier ein drittes Signal auswählen können. Dafür muss das Steuersignal für diesen Multiplexer auf 2 Bit vergrößert werden. Für den addSub Befehl muss MemToReg den Wert 10 annehmen. Damit dass Ergebnis dann auch geschrieben wird, muss zusätzlich noch RegWrite auf 1 gesetzt werden. Ins Data Memory soll nichts geschrieben werden, also wird MemWrite auf 0 gesetzt.

Veränderung der Taktlänge:
Dieser Befehl ist mit einem normalen R Typ Befehl vergleichbar. Es kommt zusätzlich noch ein Subtrahierer hinzu (125ps) und ein Multiplexer wird von 2:1 auf 4:1 erweitert (50ps statt vorher 30ps => +20ps).

Ohne diesen Befehl (Äquivalente Befehlsabfolge)

```
add $r0, $r1, $r2
sub $r0, $r0, $r3
```

Es wird also statt bisher zwei nur mehr ein Taktzyklus benötigt, also ein Taktzyklus gespart.

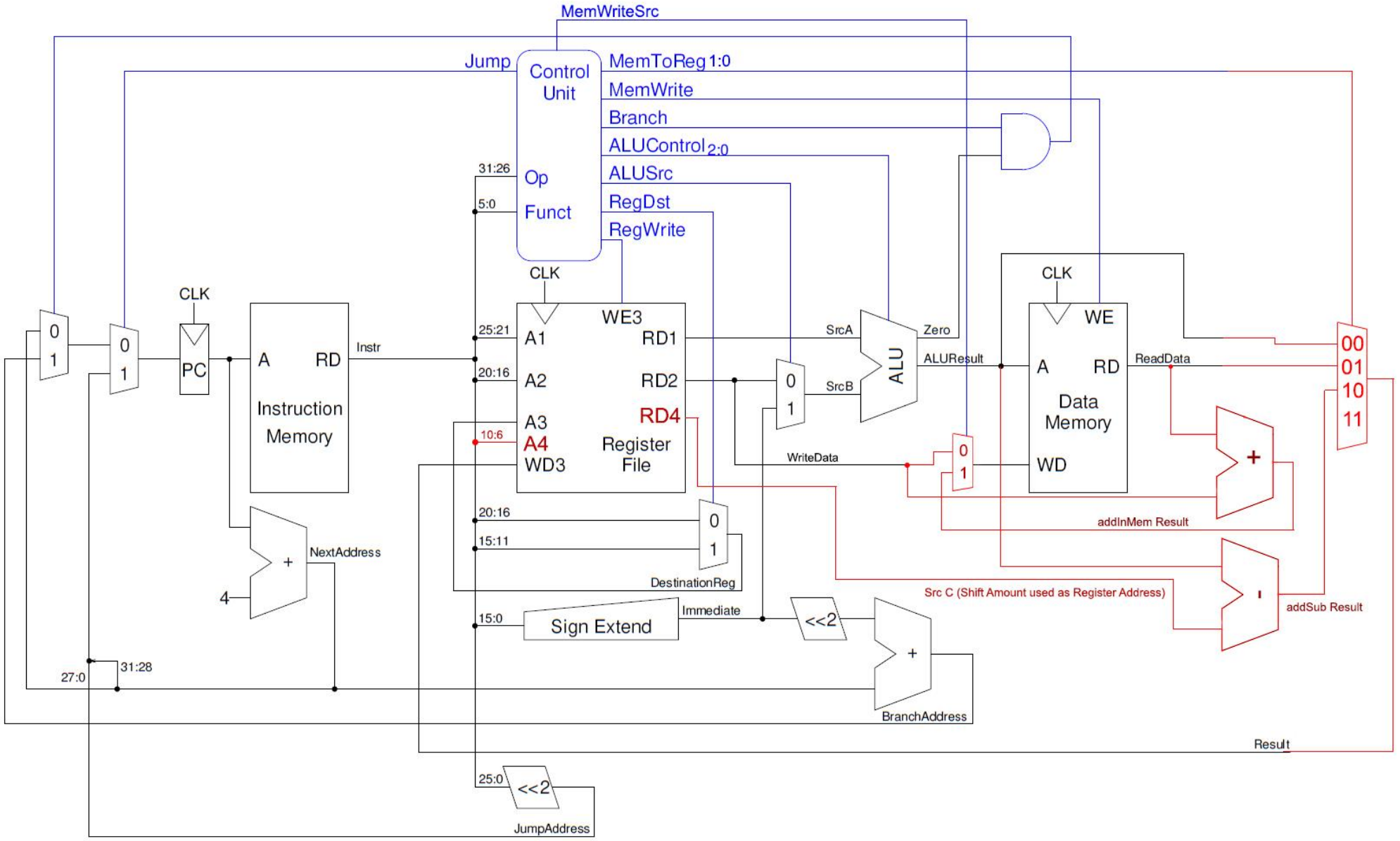

Abbildung 1 MIPS-Prozessor mit Unterstützung für addInMem und addSub