3.4. Pipelining

F6 oben normaler Befehlsablauf ohne Pipelinig, REIN SEQUENTIELLER Ablauf aller Phasen

F6 Mitte: Ablauf mit 5-stufiger Pipeline:

Nach Abschluss einer Phase für Befehl i wird diese Phase für Befehl i+1 gestartet (das gilt für alle Phasen)

Ohne Pipelinig: Befehlslänge 5 Takte, alle 5 Takte ein Befehl fertig
Mit Pipelining: Befehlslänge 5 Takte, jeden Takt ein Befehl fertig

➔ Verfünffachung der
➔ der Befehle ohne wesentlich höheren Hardwareaufwand, da im Prozessor für jede Phase andere Teile benötigt werden

F6 unten: Superpipeline (hier 10-stufig)

- Aufteilung jeder der 5 Phase je in 2 Unterphasen
- Dadurch ist höhere Taktfrequenz möglich
- Befehl ist 10 Takte lang (durch höhere Taktfrequen aber nicht länger als bei der 5-stufiger Pipeline
- Je Takt ein Befehl fertig

➔ Verzehnfachung der Leistung

Pipline-Probleme:

- Datenkonfliktbei zwei aufeinanader folgenden Befehlen, wo der zweite Daten benötigt, die der erste berechnet sind diese nach WB (1. Bef.) verügbar, würden aber 2 Takte vorher am Beginn von OF (2. Befehl) gelesen -> Fehler

Mögliche Lösung (F7 oben):

Verzögerung des OF des Folgebefehls um 2 Takte, d.h. OF vom Folgebefehl nach WB des vorherigen Befehls. Das bedeutet im Bsp. R3 ist mit dem neuen Wert geschrieben, bevor R3 wiedergelesen wird.

➔ Kein regulärer Ablauf, geringere Leistungssteigerung, zusätzliche Logik zur Erkennung der Datenkonflikte (nach jeder ID Phase ist der aktuelle Behl bzgl. des vorhergehenden und des vorvorhergehenden Befehls zu analysieren)

Weitere Lösung:
Result Forwarding (F8):
EX des Folgebehls ist nach EX des vorherigen Befehls, d.h. die vom Folgebefehl benötigten Daten sind bereits berechnet aber nicht ins Register geschrieben. Bei direktem Übergeben dieser Daten vom ALU-Ausgang zum ALU-Eingang (in F8) über Multiplexer (MUX) kann auf die Lücke verzichtet werden. Die berechneten Daten des vorherigen Befehl werden das parallel in das Register geschrieben.

Multiplexer: Logische Struktur, die von zwei (oder mehreren Datenworte) genau eins auf den Datenwortausgang schaltet.

- Sprungkonflikte

Bei bedingtem Sprung (i.a. in Abhängigkeist von einem Prozessor-Staturregisterbit (Flag)) ist der Wert des vorherigen Befehls erst nach dessen WB verfügbar, wird aber 3 Takte vorher (vor ID-Phase) des aktuellen Befehls (bedingter Sprung) benötigt.

➔ Bei nicht erfüllter Bedingung: normaler Pipelineablauf (Ergebnisse der Folgebefehle (sequentiell) sind gültig)

➔ Bei erfüllter Bedingung sind die Ergebnisse der Folgebefehle nicht gültig, d.h. es entsteht eine Lücke, die Ergebnisse dürfen nicht in die Register geschrieben werden

Nicht regulärer Ablauf, Verringerung der Leistungssteigerung

Verminderung des Problems Sprungkonflikt:

Sprungvorhersage:
Aus dem Verhalten des Spungbefehls bei der ltzen Abarbeitung wird auf das möglich Verhalten bei der aktuellen Abarbeitung geschlossen.

1-bit-Vorhersage (F9 Mitte)

Beim letzten mal gesprungen führt zur Vorhersage aktuell wird auch gesprungen.
Beim letzten mal nicht gesprungen führt zur Vorhersage aktuell wird auch nicht gesprungen

Zur Realisierung dieser Vorhersage wird ein Sprung-Vorgeschichte-Tabelle (Branch History Table) notwendig:

Dort stehen die Adressen der im aktuellen Kontext des Programms aktiven Sprungbefehle mit je einem Zusatzbit der Sprungbeobachtung bei der letzten Ausführung. Dieses bit wird ausgelesen wen der Sprungbefehl auf der zugeordnete Adresse erneut ausgeführt wird und über den Automaten wird die Vorhersage erzeugt.

Zusätzlich notwendige Hardware:
Speicher für Tabelle mit Vorhersagebit.
Vergleicher für alle Adresse in der Tabelle mit der Adresse des aktullen Sprungbefehls zum Auslesen des richtigen Vorhesagebits. Vergleicher muss parallel alle Adressen in der Tabelle mit der aktuellen Adresse vergleichen (wegen dem Zeitverbrauch)

2-bit-Sprungvorhersage: (F9 rechts)

Um zu einer Vorhersage Springen zu kommen muss der Befehl zwei mal gesprungen sein, hintereinander

Um zu einer Vorhersage Nicht Springen zu kommen muss der Befel zwei mal nicht geprungen sein, hintereinander

Aufwand: je Tabellenplatz ein bit mehr, etwas aufwendigerer Automat
Gewinn nur am Schleifenanfang korrekte Vorhersage

Branch Target Buffer (F11)
Zieladress ist zugeordnet zur Adresse des aktuellen Befehls mit in der Tabelle entsprechend der Sprungvorhersage, muss nicht gelesen werden, wenn die Vorhersage korrekt ist.

Hardwareaufwand: Tabelle wird in der Datenbreite verdoppelt.