Konrad Zuse und der bedingte Sprung

Raúl Rojas

Es ist gut bekannt, dass keine der ersten von Konrad Zuse gebauten Rechenmaschinen den bedingten Sprung im Befehlssatz aufwies. Sowohl die Z1 (1936–1938), die Z3 (1939–1941), als auch die Z4 (1941–1945) wurden ursprünglich für feste Folgen von arithmetischen Rechenoperationen konzipiert [1], d. h. die Maschinen konnten Addition, Subtraktion, Multiplikation, Division und einige andere arithmetische Befehle sequenziell bearbeiten. Im Zusammenspiel mit einem Speicherwerk konnten damit komplexe arithmetische Ausdrücke ausgewertet werden. Befehle für die Ein- und Ausgabe waren vorhanden – einzig der bedingte Sprung fehlte, um die Z3 und Z4 als vollwertige Computer gelten lassen zu können [2].

Die Programme für die Zuse-Rechner wurden auf einem binären Lochstreifen untergebracht. Die Befehle wurden nacheinander abgerufen und ausgeführt. Das Programm konnte beim Lauf der Maschine nicht verändert werden – man konnte höchstens den Lochstreifen zu einer Schleife binden und es wiederholt ausführen lassen. Ich habe deswegen an anderer Stelle gezeigt, dass auch ohne den bedingten Sprung ein Universalrechner realisiert werden kann, indem mit einer solchen arithmetischen Schleife der IF-Befehl simuliert wird [3]. Das Ganze ist zwar theoretisch vorstellbar, aufgrund der exzessiven Aufblähung des Programmcodes (und damit der Länge des Lochstreifens) jedoch praktisch undurchführbar.

Der bedingte Sprung wurde schließlich in der Z4 eingebaut, bevor Zuse diese Maschine an die ETH-Zürich als Miet-Kauf übereignen konnte [4, 5]. Auf Wunsch der ETH-Mathematiker wurde der Befehlssatz an die Bedürfnisse der Numerik angepasst: d. h. der bedingte Sprung und weitere bedingte Befehle kamen hinzu. Es war damit möglich, die Logik der Berechnung adaptiv zu den Resultaten zu ändern, wie heute bei jeder Programmiersprache üblich.

Dieser Aufsatz beschreibt die Implementierung des bedingten Sprungs bei der Z4, dem ersten kommerziellen Computer auf dem europäischen Festland (Großbritannien überlassen wir seinen *splendid isolation*). Der Aufsatz stützt sich auf eine Dokumentation einer Ausstellung an der ETH von 1981, die mir von Herbert Bruderer freundlicherweise übermittelt wurde [6]. Wir wissen mit Sicherheit, dass bedingte Befehle in der Z4 erst nach 1945 eingeführt wurden [7].

Für die folgende Beschreibung genügt es zu wissen, dass der Prozessor der Z4 zwei Gleitkomma-Register besaß (OR-I und OR-II genannt) und 64 Speicherzellen. Alle arithmetischen Operationen benutzten diese Register und legten das Resultat in OR-I ab.

Die Lochstreifen der Z4

Für die ETH wurden zwei Lochstreifenleser verwendet (Zuse hatte an noch mehr Abtaster gedacht, sie aber vorläufig nicht realisiert). In dem ersten (At_0, für Abtaster Null) konnte das Hauptprogramm ausgeführt werden, bis der Befehl „Up“ die Steuerung an den zweiten Lochstreifen (At_1, für Abtaster Eins) übergab, wie wir weiter unten besprechen. Damit war es möglich, Unterprogramme aus einem Hauptprogramm aufzurufen.

DOI 10.1007/s00287-013-0717-9

Raúl Rojas
FB Mathematik und Informatik, Freie Universität Berlin,
Arnimallee 7, 14195 Berlin
E-Mail: rojas@inf.fu-berlin.de

Published online: 03 July 2013

Zusammenfassung

Dieser Aufsatz beschreibt die von Konrad Zuse für die Rechenmaschine Z4 eingeführten bedingten Befehle, inklusive dem bedingten Sprung. Die extra für numerische Berechnungen angedachte Anweisungen wurden um 1950 auf Bitte der Mathematiker der ETH-Zürich in der Z4 eingebaut. Der bedingte Sprung ignorierte alle nachfolgende Befehle im Programmlochstreifen bis zur „Start"-Marke im Code, falls der Wahrheitswert im ersten Arbeitsregister auf wahr gesetzt war. Anderenfalls wurde der Sprung ignoriert. Einfache Programmierbeispiele erleichtern das Verständnis der Funktionsweise der bedingten Befehle der Z4.

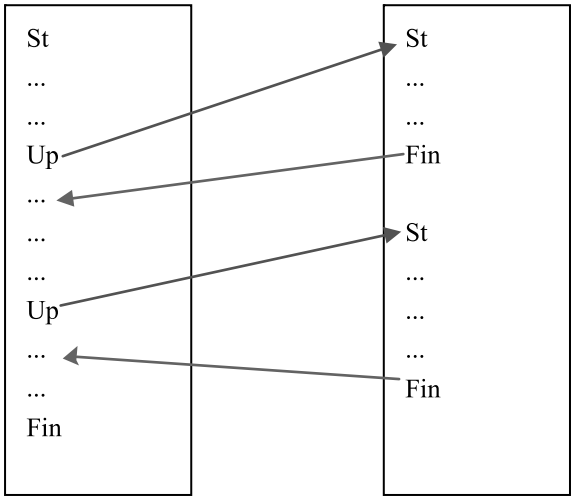

Abb. 1 Unterprogrammaufrufe. Der Abtaster At_0 (links) übergibt die Steuerung zweimal an den Abtaster At_1 (rechts). Die Unterprogramme werden in der Reihenfolge ihrer Präsenz im Lochstreifenleser verwendet.

Der Programmcode fing mit dem Befehl „St" an. Das Programm endete mit dem Befehl „Fin". Beim Erreichen von Fin im Hauptprogramm wurde eine rote Lampe eingeschaltet, sodass der Operator das Ende des Programms erkennen konnte.

Der Befehl Up übergab also die Steuerung vom Hauptprogramm in At_0 zum Unterprogramm in At_1. Beim Wechsel wurde der Lochstreifen in At_1 bis zum nächsten St-Befehl fortgespult. Die neue Befehlsfolge kam dann aus At_1. Beim Erreichen von Fin wurde die Steuerung zurück an At_0 gegeben. Unterprogrammparameter konnten über jede Speicherzelle übertragen werden, so wie auch die Resultate (es gab nur einen globalen Speicher von Zellen mit Adressen 0 bis 63). Allerdings wurde die Adresse 63 für die Übermittlung eines Fehlercodes verwendet. Da die Z4 mit Gleitkommazahlen arbeitete, konnten arithmetische Ausnahmen in Fehlerbits festgehalten werden (unendliches Resultat, oder Not-a-Number), wie beim heutigen IEEE-Floating Point-Format. Nach der Rückkehr aus einem Unterprogramm konnte Adresse 63 getestet werden. Der Z4-Befehl „$x =?$" prüft, ob der Inhalt der Adresse 63 (vorher in OR-I geladen) eine zulässige Gleitkommazahl ist. Damit konnten arithmetische Ausnahmen in Unterprogrammen an das Hauptprogramm gemeldet werden. Abbildung 1 zeigt ein Diagramm der Ausführungsreihenfolge bei zwei Unterprogrammaufrufen. Die Unterprogramme waren allerdings nicht durchnummeriert. Die Reihenfolge ihrer Ausführung ergab sich aus der aktuellen Position des Lochstreifens in At_1.

Bedingte Befehle

Zuse hat für die Z4 Wahrheitswerte vorgesehen, und zwar – 1 für falsch und + 1 für wahr. Es gab logische Operationen, die den Wahrheitswert in OR-I ablegten. So legt z. B. die Operation „$x = 0$" ein + 1 in OR-I falls OR-I gleich Null ist, ansonsten – 1. Weitere logische Operationen waren z. B. der Vergleich von OR-I mit 1, bzw. OR-I größer-gleich Null. Mit den resultierenden Wahrheitswerten konnten dann bedingte Befehle gelenkt werden.

Der Befehl Up′ (mit Quote-Zeichen) war äquivalent zu Up, wurde aber bedingt ausgeführt. War der Wahrheitswert in OR-I falsch, wurde Up′ übersprungen; war OR-I wahr, so wurde die Steuerung doch an das Unterprogramm in At_1 übergeben (Abbildung 2).

Der Befehl Fin′ *im Hauptprogramm* war äquivalent zu Fin, wurde aber nicht ausgeführt, wenn der augenblickliche Wahrheitswert in OR-I falsch war.

Fin′ *in einem Unterprogramm* wurde ebenfalls nicht ausgeführt, falls OR-I falsch war. War OR-I aber wahr, dann stoppte die Maschine vollständig. In diesem Sinne war ein ausgeführtes Fin′ in Unterprogramm, wie ein Fin in Hauptprogramm. Diese Asymmetrie der Fin-Befehle war in der Z4 nicht angenehm, Zuse muss aber Gründe dafür gehabt haben.

Abstract

This paper describes the conditional instructions retrofitted by Konrad Zuse to the instruction set of the Z4 around 1950, including the conditional jump. The instruction set upgrade for the Z4 was a request from the mathematicians of the ETH Zürich, who wanted the increased functionality mainly for numerical computations. In case that the truth value in a working register was true, the conditional jump ignored all instructions in the punched tape containing the program, one after another, until a "start" mark in the code was reached. The jump instruction was otherwise ignored. Some simple programming examples help to understand the operational semantics of the conditional instructions of the Z4.

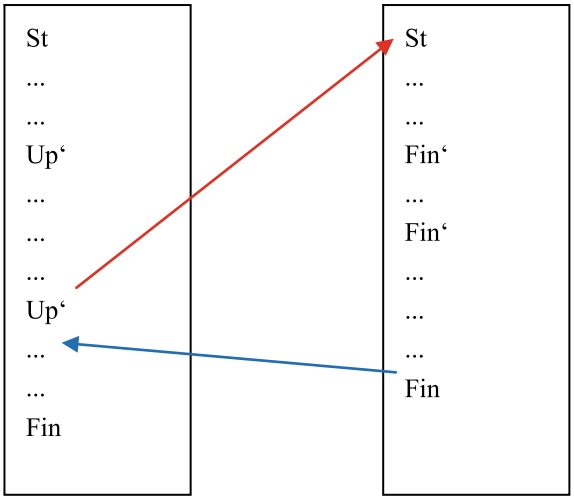

Abb. 2 Unterprogrammaufruf. Der Abtaster At_0 übergibt die Steuerung an den Abtaster At_1. Der erste Up‘ wird nicht ausgeführt (OR-I enthielt – 1), nur der zweite Up‘ wird ausgeführt. Die erste beide Fin‘ werden nicht ausgeführt (da OR-I in dem Augenblick – 1 enthielt). Erst der Befehl Fin am Ende übergibt die Steuerung an das Hauptprogramm zurück.

Natürlich sollten die Befehle Up und Up′ in Unterprogrammen in At_1 nicht verwendet werden, da es keine weitere untergeordnete Abtaster in der Z4 gab.

Der bedingte Sprung

Und schließlich: der bedingte Sprung. Es war eine Odyssee von fast vierzehn Jahren (1936–1950), bis der erste bedingte Sprung in einer Zuse-Maschine

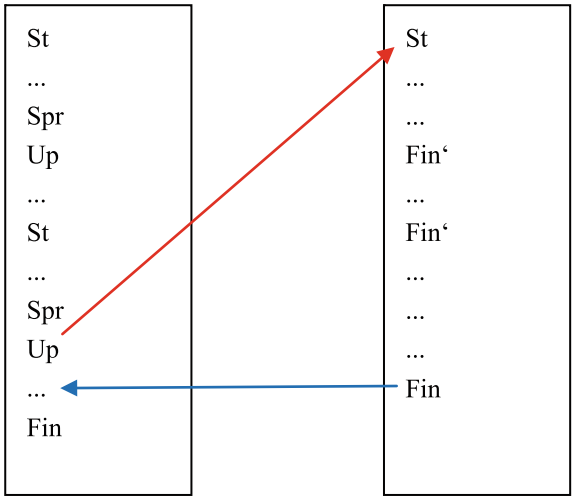

Abb. 3 Der erste Befehl Up wird übersprungen, der zweite nicht. Im Unterprogramm werden die zwei Befehle Fin‘ nicht ausgeführt. Die passende Wahrheitswerte in OR-I dafür sind: wahr, falsch im Hauptprogramm, und falsch, falsch im Unterprogramm.

das Licht der Welt erblickte. Der Z4-Befehl „Spr“ in einem Programm oder Unterprogramm bewirkte, *dass alle nachfolgenden Befehle bis zum nächsten Start-Befehl* (St) *übersprungen wurden* (Ambros Speiser nannte diesen Befehl Spr′, anders als im Programmierheft der Z4 [8]). D. h. das Befehlspaar Spr-St wirkt wie eine Klammer: alles dazwischen wird nur bedingt ausgeführt, falls das Register OR-I eine – 1 enthält, ansonsten werden all die eingeklammerte Befehle übersprungen (Abbildung 3).

Weitere Feinheiten der Befehlsfolgen sind für uns nicht relevant (es gab in der Z4 verbotene Befehlskombinationen). Bleibt nur zu unterstreichen, wie einfach die Lösung von Zuse für die Einführung von bedingten Befehlen war. Der bedingte Sprung bezieht sich nicht auf Adressen (es ist kein GOTO *n*), d. h. das Programm kann nicht „rückwärts“ im Lochstreifen springen, nur nach vorne, bis zum nächsten St-Pseudobefehl. Die Befehle Up′ und Fin′ sind ähnlich einfach implementierbar: man kann sie je nach Wahrheitswert in OR-I ignorieren. Es lässt sich aber mit solchen zusätzlichen Befehlen alles Mögliche programmieren.

Wir haben oben angemerkt, dass die Unterprogramme in der Reihenfolge ihrer Präsenz in Lochstreifen beim At_1 aufgerufen werden. Wollen wir aber den Unterprogrammen Kennungen vergeben, sodass wir z. B. Unterprogramm 5 oder 6

aufrufen können, ist dies einfach zu realisieren. Dafür schließt man den Unterprogrammlochstreifen zu einer Schleife. In Adresse 0 wird als Zahl abgelegt, welches Unterprogramm gewünscht ist (5 bzw. 6). Am Eingang des Unterprogramms 5 kann man Adresse 0 in OR-I laden, 5 subtrahieren, und OR-I auf null testen. Ist das Resultat falsch, überspringen wir alles bis zum nächsten St-Zeichen, der Anfang von Unterprogramm 6. Und umgekehrt: Unterprogramm 6 testet, ob Adresse 0 eine 6 enthält, usw. (allerdings dürfen die Unterprogramme keine Spr-Befehle verwenden, sonst wird es komplizierter). Man durchschaut aber die Idee: Mit den bedingten Befehlen lässt sich viel flexibler programmieren.

Die Konkurrenz

Heute, fast 70 Jahre nach 1945, mutet es als Kuriosum an, dass Konrad Zuse so spät auf die Implementierung des bedingten Sprungs kam. Man darf aber die damaligen Möglichkeiten und sogar die Arbeit der Konkurrenz nicht außer Acht lassen. Auch die amerikanische ENIAC, häufig als erster elektronischer Rechner der Welt gepriesen, besaß am Anfang keinen bedingten Sprung [9]. Dieser wurde erst im Nachhinein durch einen Trick realisiert: Datenleitungen wurden als Signalleitungen zweckentfremdet, und so konnte ein Vorzeichenbit eine Kette von Rechenoperationen starten. Vorgesehen war dies am Anfang des Designs nicht. Auch die Maschine von John Atanasoff war nur für eine feste Folge von Operationen vorgesehen (der Gauß-Algorithmus für die Lösung von linearen Gleichungen). In der Turing-Maschine von 1936 sind Sprünge implizit in den Zustandswechseln eingebaut. Wenn aber die Universelle Turing-Maschine eine spezifische Turing-Maschine simuliert, müssen bei bedingten Operationen (und nicht nur bei diesen!) auch Programm und Datenzellen mühsam eins nach dem anderen übersprungen werden.

Es ist deshalb in einem solchen zeitlichen Abstand manchmal einfach die Kurzsichtigkeit von Erfindern zu bemängeln. Man muss aber bedenken, dass jene Forscher damit beschäftigt waren, etwas völlig Neues zu erschaffen. Der Computer kam nicht in die Welt wie Athena aus dem Kopf von Zeus, vollendet und mit Rüstung. Es gab nicht einfach einen Gedankenblitz – die Entwicklung des Computers war eine ausgedehnte revolutionäre Epoche, die sich bis heute fortsetzt.

Literatur

1. Rojas R (1998) (Hrsg) Die Rechenmaschinen von Konrad Zuse. Springer, Berlin
2. Rojas R (1997) Konrad Zuse's legacy: the architecture of the Z1 and Z3. Ann Hist Comput 19(2):5–16
3. Rojas R (1998) How to make Konrad Zuse's Z3 a universal computer. IEEE Ann Hist Comput 20(3):51–54
4. Zuse K (1993) Der Computer – Mein Lebenswerk, 3. Aufl. Springer, Berlin
5. Bruderer H (2012) Konrad Zuse und die Schweiz: Wer hat den Computer erfunden? Oldenbourg Wissenschaftsverlag, München
6. Anonym (1981) Dokumentation: Konrad Zuse und die Frühzeit des wissenschaftlichen Rechnens an der ETH (ETH Zürich, 17.6.–15.7.1981). Die Dokumentation enthält die Liste der Befehle für die Z4
7. Zuse K (ca. 1945) „Rechenpläne für das Rechengerät V4". Zuse Papers, GMD 011/006. Online vorhanden im Zuse Internet Archiv, http://www.zib.de/zuse/Inhalt/Texte/Chrono/40er/Pdf/329scan.pdf, letzter Zugriff 30.6.2013
8. Speiser A (2008) Episoden aus den Anfängen der Informatik an der ETH. Informatik-Spektrum 31(6):606–612
9. Goldstine H (1996) The Electronic Numerical Integrator and Computer (ENIAC). Ann Hist Comput 18(1):10–16