MERAC Evaluator

Rainer Glaschick, Paderborn, Germany
email: rainer@glaschick.de
http://rclab.de/
2017-09-21

1. Zusammenfassung

MERAC steht für *Mechano-Elektrischer Retrograder Automatischer Computer* und bezeichnet Entwürfe für einfache programmierbare Rechner, die durch meine Beschäftigung mit der *Analytical Engine* (AE) von Babbage und dem ENIAC sowie weiteren frühen Rechnern entstanden sind und zur deren Erklärung nützlich sind.

Der vorführbare Testaufbau mit zwei dreistelligen Dezimalregistern dient der ersten Erprobung der Komponenten. Die Programmierung verwendet 7-Bit Programmelemente; der Programmablauf wird vom Benutzer durchgeführt. Ein Konzept für die automatische Programmausführung einschliesslich strukturierter bedingter Ausführung und Wiederholung ist fertig, aber noch nicht aufgebaut.

Bereits damit wird deutlich, dass spätestens zu Beginn des 20. Jahrhunderts, als Fernmelderelais in großen Stückzahlen verfügbar waren, eine programmierbare elektro-mechanische (Dezimal-)Rechenmaschine mit wenig Aufwand hätte gebaut werden können.

Vergleiche mit den genannten und anderen frühen Rechnern sind am Schluss gesammelt

dargestellt.

2. Architektur des dezimalen MERAC

Die Maschine besteht aus einer Reihe von dezimalen Registern, wobei ein Programm die Inhalte abrufen oder ersetzen kann. Die Ziffern werden als Rotationswinkel einer Scheibe gespeichert.

2.1. Komponenten

Die Komponenten sind in folgender Grafik für zwei Register dargestellt:

Ein Taktgenerator (`CG`) sendet Taktimpulse an den Programm-Sequenzierer (`PS`). Dieser selektiert (`Sel`) die additiven Register (`AM1` und `AM2`) entsprechend der Information in einer (Mikro-) Programmzeile und gibt gemeinsame Signale (`Clock`) weiter und steuert den Modifikator (`MU`); letzterer koppelt den Ein- und Ausgabeteil des Ziffernbusses (`Digit Bus`).

Eine Zahl, die von einem Register ausgelesen wird, erzeugt für jede Ziffer genau die Anzahl von Impulsen, die ihrem Stand entspricht, und schickt diese an das Zielregister. Da die Speicherung der Ziffernwerte durch Zählscheiben erfolgt, ist es naheliegend, die empfangenen Impulse einfach zu dem Zähler addieren zu lassen. Dann ist noch ein Übertrag auf die nächste Stelle zu bewirken, wenn eine Ziffer von 9 auf 0 gewechselt hat.

Das Auslesen eines Registers erfolgt durch eine vollständige Umdrehung in Vorwärtsrichtung und dem Senden der Anzahl der dazu notwendigen Impulse nach dem Übergang von 9 auf 0, der ohnehin für den Übertrag benötigt wird.

Es gibt keine Operation, die den vorherigen Wert eines Registers unmittelbar überschreibt, da dazu das Register zunächst auf Null zu stellen wäre und das die Maschine, da sie mit fester Zykluszeit arbeitet, wesentlich verlangsamen oder vergrößern würde. Vielmehr kann das Quellregister zusätzlich beim Auslesen gelöscht werden, indem es beim Erreichen der Nullstellung nicht weitergedreht wird.

Die Übertragung der Ziffernimpulse erfolgt über einen Bus, also eine Sammelleitung pro Ziffer. Wird ein Register als Quelle gewählt, sendet es die entsprechenden Impulse auf die entsprechende Ziffernleitung, und das Zielregister übernimmt die Impulse. Alle nicht aktivierten Register sind von den Busleitungen getrennt bzw. ignorieren die Impulse.

Entgegen anderer Maschinen sind der Ein- und Ausgabe-Bus getrennt und durch einen Modifikator verbunden. Bei einer Addition sind die jeweiligen Busleitungen direkt miteinander verbunden. Für eine Subtraktion wird immer dann ein Impuls auf der Leitung zum Zielregister erzeugt, wenn von dem Quellregister kein Impuls geschickt wird, so dass die Summe der empfangenen und gesendeten Impulse 9 ergibt, also das 9-er Komplement erzeugt wird. Steht also 00123 im Zielregister und sendet das Quellregister 00021, dann erzeugt der Modifikator die Zahl 99978, und das Zielregister steht dann, da der Überlauf der obersten Stelle verloren geht, auf 00101 und ist somit um 1 kleiner als das Ergebnis. Daher wird im Fall einer Subtraktion an der niedrigsten Stelle — die ja eigentlich keinen Übertrag bekommen kann — ein Übertrag simuliert, so dass das Ergebnis 102 ist.

Zusätzlich zu einer Modifikation der übertragenen Zahl kann der Modifikator auch bestimmen, ob die übertragene Zahl Null oder negativ war, um eine bedingte Verzweigung im Programm zu erlauben.

Um Multiplikationen per Software ausführen zu können, kann der Modifikator die Dezimalstellen verschieben, indem die Busleitungen vertauscht werden.

Wenn Quell- und Zielregister im Befehl binär codiert werden und alle Register vorhanden sind, umfasst das Programm ein weiteres Bit für den Modifikator, um eine Null substituieren, wenn ein Register inkrementiert werden oder nur die Sprungbedingung evaluiert werden soll.

Der Wert einer Ziffer wird physisch nicht als einzelne Impulse auf dem Bus übertragen;

vielmehr wird ein Freigabsignal aktiviert und dann im Register in einzelne Impulse mittels eines globalen Takts gewandelt, der ohnehin für das Auslesen benötigt wird.

2.2. Automatische Programmsteuerung

Die für den dezimalen MERAC mit getrenntem sequentiellem Programmträger geplante Programmsteuerung arbeitet folgendermaßen:

Die Pilotversion hat 16 Register. Die Anzahl der Dezimalziffern wird nur durch die Datenbreite des Busses und des Modifikators bestimmt; sie ist unabhängig von der Bitzahl im Befehl. Zur Erprobung sollten 5 Ziffern ausreichend sein.

Befehle zum Datentransfer und Steuerbefehle werden durch ein Bit unterschieden und haben unterschiedliche Aufteilung in Teilfelder.

Datentransferbefehle enthalten die beiden Registernummern binär (4 Bit) codiert. Die beiden Bits `L` (Löschen) und `E` (Einerübertrag) werden weiterhin benötigt und an alle Register geleitet, aber nur von dem betroffenen Register ausgewertet.

Sodann gibt es Bits für den Modifikator, der folgende Operationen auf dem Datenbus erlaubt:

```
Z Zero Datentransfer unterdrücken
N Neuerkomplement bilden
M Multipliziere mit 10 (durch Vertauschen)
D Dividiere durch 10
```

Somit umfasst das Befehlswort mindestens 1+4+4+2+4 = 15 Bit.

Die Steuerbefehle haben folgende 6 Steuerbits:

```
V  Vorwärts (sonst Rückwärts) bewegen
I  Invertierte Bedingung
Z  Wenn Zero Flag gesetzt
N  Wenn Negative Flag gesetzt
H  Höchste Ziffer war null
L  Niedrigste Ziffer war null
```

Damit bleiben 8 Bit, um das Sprungziel zu bestimmen.

Weil als Programmträger ohnehin ein serielles Medium (Lochstreifen bzw. Lochkarten) verwendet wird, ist die Zieladresse keine Distanz in Befehlen, sondern einfach ein Bitmuster. Wird der sequentielle Ablauf verlassen, dann wird das Programmband solange bewegt, bis ein Sprungbefehl mit demselben Bitmuster gefunden wird; wobei eine Rückwärtsbewegung nur nach einem Vorwärtssprung sucht, und umgekehrt; der so gefundene Befehl wird nicht ausgeführt.

Die Bewegung des Programmträgers kann im Takt der Ziffernimpulse erfolgen, also zehnmal schneller als ein Datenbefehl.

Die Anwendung sollte in strukturierten Schleifen erfolgen, d.h. jede Schleife beginnt mit einem bedingten Vorwärtsprung und endet mit einem Rückwärtssprung mit der invertierten Bedingung. Als Bitmuster wird dann die Verschachtelungstiefe verwendet.

2.3. Ein- und Ausgabe

Eine Ausgabe von Resultaten könnte erfolgen, indem ein Register so aufgebaut wird, dass die Ziffernscheiben mit erhabenen Ziffern versehen sind und über ein Farbband auf Papier gedruckt werden (Tischrechner).

Eingabe erfolgt über ein Register, das von einem Lochstreifen oder von Lochkarten mit Ziffern vorbelegt wird.

In beiden Fällen muss der Programmablauf angehalten werden, bis die Ein- oder Ausgabe bereit sind.

Anstatt über ein Register könnte eine Ein- oder Ausgabe auch über den Modifikator erfolgen.

2.4. Unterprogramme

Dynamische Unterprogramme sind mangels Direktzugriffsspeicher nicht verfügbar. Für eine (vorzeichenlose) Multiplikation werden etwa 10 Befehle benötigt; sie kann also als vorbereitete Befehlsfolge eingefügt werden.

2.5. Speicherbedarf

Als typische Aufgabe für einen Rechner dieser Klasse kann die lineare Regressionsanalyse dienen (beste Gerade).

Die einzugebenden Wertepaare (x_i, y_i) seien auf Lochkarte oder Lochstreifen vorhanden; es sind die folgenden vier Summen zu berechnen:

$\Sigma\ x_i$

$\Sigma\ y_i$

$\Sigma\ x_i * x_i$

$\Sigma\ x_i * y_i$

Das ergibt folgende Anzahl von Registern:

```
Eingabewerte x und y: 2
Multiplikation: 3
Summen: 4
```

also 9 Register. Dabei kann y in einem Multiplikanden geführt werden, da es sogleich zur Summe addiert werden und dann bei der Multiplikation aufgebraucht werden kann, so dass minimal 8 Register benötigt werden.

Die beiden Multiplikationen benötigen nicht zwingend Unterprogramme, sondern können als fertig vorbereitete Abschnitte eingefügt werden.

Die Berechung des Regressionskoeffizienten aus den Summen kann, wenn der Programmieraufwand nicht gerechtfertigt ist, auch mit einem Tischrechner erfolgen. Selbst bei manueller Eingabe der Werte wäre ein solcher Tischrechner eine wesentliche Entlastung.

3. Der Evaluator

Der MERAC-Evaluator ist die kleinste Machine, mit der eine erste Überprüfung der Komponenten möglich ist und die die Elemente des Konzepts veranschaulicht werden können.

3.1. Aufbau

Der Evaluator umfasst:

- zwei Register mit je drei Ziffern
- einen elementaren Modifikator
- eine manuelle Programmsteuerung

Drei Ziffern pro Register sind ausreichend und notwendig, um den Übertagsmechanismus zu erproben.

Die manuelle Programmsteuerung besteht aus einer auswechselbaren Matrix, deren Zeilen durch einen Drehschalter angewählt werden können und durch eine Taste einmal aktiviert werden. Der Ablauf wird also durch den Bediener manuell gesteuert. (Allein schon dieses

wäre — bei mehr und größeren Registern — ein Vorteil gegenüber den herkömmlichen Tischrechnern gewesen.)

Jedes (Mikro-) Programmwort besteht aus 7 Bits:

- A1: Zu Register 1 addieren
- R1: Register 1 auslesen
- A2: Zu Regster 2 addieren
- R2: Register 2 auslesen
- L: Lösche Register durch das Lesen
- E: Einerübertag setzen
- N: Neuner-Komplement bilden

Der Übersichtlichkeit halber werden die nicht gesetzten Bits einer Zeile durch einen Punkt und ein gesetztes Bit durch einen Buchstaben dargestellt.

Um Register 1 zu Register 2 zu addieren, ist die Zeile wie folgt zu setzen:

```
.R A. ...
```

Zum Subtrahieren in der Gegenrichtung ist notwendig:

```
A. .R .EN
```

Um Register 1 um 1 zu erhöhen, wird folgende Zeile verwendet:

```
A. .. .E.
```

Um ein Register zu löschen, wird es ausgelesen, aber nicht zu dem anderen addiert:

```
.R .. L..
```

Da so eine Addition von 1 ohne ein Hilfsregister möglich ist, wurde das E-Bit separat zugänglich gemacht und nicht mit dem N-Bit verkoppelt.

Ist kein Quellregister angesprochen, wird eine Null gesendet; durch Setzen des `N`-Bits also die Zahl `999`, so dass Register 1 um eins vermindert wird durch:

```
A. .. ..N
```

Der elementare Modifikator kann nur das 9-er Komplement bilden (und weder eine Ziffernverschiebung noch eine Vorzeichenbestimmung durchführen).

3.2. Negative Zahlen

Es könnte sowohl 9-er oder 10-er Komplement benutzt werden; bei dem letztern ist bei Subtraktionen zusätzlich zur Bildung des 9-er Komplementes in der MU auch ein Inkrement durchzuführen.

Die Auswahl betrifft einerseits die Ein- und Ausgabe, hier an den Ziffernrollen dargestellt:

Die erste Stelle wird voll ausgenutzt (nicht nur alternierend mit + und -), deren Beschriftung lautet daher:

```
0  1  2  3  4  -4  -3  -2  -1  -0
```

wobei statt des Minuszeichens die Ziffern in roter Farbe sind.. Die anderen Ziffernrollen zeigen dann:

```
0|-9  1|-8  2|-7  3|-6  4|-5  5|-4  6/-3  7|-2  8|-1  9|-0
```

Dies gilt für die Verwendung des 9-er Komplements auch für die letzte Stelle.

Beim 10-er Komplement erhält die letzte Stelle folgende Beschriftung:

```
0|-9+  1|-9  2|-8  3|-7  4|-6  5|-5  6|-4  7|-3  8|-2  9|-1
```

Das ergibt für die Anzeige folgender negativer Zahlen im 10er Komplement die Anzeige der schwarzen Ziffern in der zweiten und der roten Ziffern in der dritten Spalte:

```
-001  999   001
-011  989   011
-010  990   009+
-456  544   456
-499  501   499
-500  500   499+
```

Ein + in der letzten Spalte zeigt an, dass die gesamte Zahl um eins zu erhöhen ist.

Die Wahl des Komplements betrifft die Multiplikation und Division sowie die MU, da es im 9er Komplement zwei Darstellungen für die Zahl Null gibt (`000` und `999`) gibt.

Bei Tischrechnern ist es üblich, durch ein verschiebliches Fenster entweder nur die schwarzen oder, bei negativen Zahlen, nur die roten Ziffern anzuzeigen. Dies wäre auch beim MERAC kein Problem.

3.3. Festkomma-Dezimalbrüche

Anstatt mit ganzen Zahlen zu rechnen, wird bevorzugt eine Festkommadarstellung mit dem Komma nach der höchsten Stelle, d.h. von -5.000 bis +4.999, verwendet.

Für Additionen und Subtraktionen ist hierzu keine Änderung notwendig; leidiglich bei Multiplikationen und Divisionen ist das Unterprogramm anzupassen.

Einfach Beispiele für Festkomma-Zahlenfolgen ohne Multiplikation und Division liegen bislang nicht vor.

3.4. Programmbeispiele

Die Beispiele sind, da nur zwei Register vorhanden sind, extrem einfach. Sie sind aber einfacher durchzuführen als auf einem vor einem halben Jahrhundert noch üblichen Tischrechner.

Die beiden Register werden mit a und b bezeichnet.

3.4.1. Dreieckszahlen

Dreieckszahlen sind die Summe der natürlichen Zahlen, also $b_{n+1} = b_n + (n+1)$

Es wird zweckmäßig mit `a=0` und `b=0` begonnen; das jeweilige Argument ist in `a`, das Resultat in `b`:

```
A. .. .E.       erhöhe a um 1
.R A. ...       addiere zu b (Ergebnis)
```

Indem die beiden Zeilen abwechselnd ausgeführt werden, wird jeweils in der zweiten Zeile die nächste Dreieckszahl erzeugt.

Die Register können beide durch einen einzigen Befehl gelöscht werden, der nicht wiederholt wird:

```
.R .R L..       Löschen von a und b gleichzeitig
A. .. .E.       erhöhe a um 1
.R A. ...       addiere a zu b (Ergebnis)
```

3.4.2. Fibonacci-Zahlen

Die Fibonacci-Zahlen sind die Summe der beiden vorherigen; man muss also nur die akutelle zu der vorherigen addieren:

$$x_{n+1} = x_n + x_{n-1}$$

Hier werden die Register abwechselnd benutzt. Anfangswerte sind a=1 und b=0:

```
.R .R L..       a=0, b=0
A. .. .E.       a=1
.R A. ...       b =+ a
A. .R ...       a =+ b
```

Zunächst sind die ersten beiden Zeilen für die Anfangswerte und dann abwechselnd die dritte und vierte auszuführen; dann erscheint die Zahlenfolge abwechselnd in beiden Registern.

3.4.3. Quadratzahlen

Multiplikationen sind nicht notwendig, um Quadratzahlen aufzuzählen.

Verwendung der ersten binomischen Formel ergibt:

$$(x+1)^2 = x^2 + 2x + 1$$

Das Argument ist in a, die Folge der Quadratzahlen in b. Die ersten beiden Zeilen setzen die Anfangswerte:

```
.R .R L..       a=0, b=0
A. A. .E.       a=1, b=1
.R A. ...       b =+ a
.R A. ...       b =+ a
A.      . ..1   a =+ 1
.. A. ..1       b =+ 1
```

Die Zeilen 3 bis 6 sind zu wiederholen; in b ist dann jeweils das Quadrat von a.

Man kann auch das Argument als erstes erhöhen, das ergibt:

$$(x+1)^2 = x^2 + 2(x+1) - 1$$

Das Programmbeispiel ist dem Leser überlassen.

Zum Aufzählen von Quadratzahlen ist die Differenzen-Methode besser. Sie beruht darauf, dass die Differenz zweier Quadratzahlen immer ungerade ist und mit jedem Schritt um 2 zu erhöhen ist:

$$(x+1)^2 - x^2 = 2x + 1$$

Das ergibt:

```
.R .R L..       a =: b, b =: 0
A. A. ..1       a =+ 1, b =+ 1
A. .. ..1       a =+ 1
A. .. ..1       a =+ 1
.R A. ...       b =+ a          Resultat ist in b
```

3.4.4. Echternacher Super-Springprozession

Es geht immer vor und eins weniger zurück, wobei die Abstände zunehmen (daher *Super*):

```
.R .R L..       a=0, b=0
A. A. .E.       a=1, b=1
.R A. ...       b =+ a
A. .. .EN       a =- 1
.R A. .EN       b =- a
A. .. .E.       a =+ 1
A. .. .E.       a =+ 1
```

Wie vor dienen die ersten beiden Zeilen der Initialisierung.

3.4.5. Quadratwurzel

Um eine Quadratwurzel zu bestimmen, kann man die Quadratzahlen aufzählen und anhalten, wenn das nächst größere Quadrat gefunden ist. Zweckmäßig verwendet man die Differenzen-Methode und subtrahiert nacheinander 1, 3, 5, 7, usw., bis das Ergebnis Null oder negativ ist. Die Anzahl der Schritte ist dann die Wurzel.

Hierbei ist das Argument im 2. Register `b` manuell nach dem Ausführen der ersten beiden Zeilen einzustellen:

```
.R .R L..       a=0, b=0
A. .. .E.       a=1         Wert für b manuell einstellen
.R A. .EN       b =- a
A. .. .E.       a =+ 1
A. .. .E.       a =+ 1
```

Eigentlich wird hier ein drittes Register benötigt; hilfsweise ist am zweiten Register ein Zähler angebracht, der die Anzahl der Operationen zählt.

4. Vergleiche mit anderen Rechnern

Eine Motivation beim Entwurf des MERAC war es, ihn mit anderen Maschinen vergleichen zu können.

4.1. Analytical Engine (1843)

Das Ziel des Mathematikers Babbage war es, Tabellen (Logarithmen, Navigationsdaten, etc) fehlerfrei berechnen zu können. Daher war bereits in den ersten Skizzen ein Druckwerk und eine Normierung, d.h. Digitalisierung, der Ziffern vorgesehen, die über den Rotationswinkel einer Scheibe dargestellt wurden. Der gesamte Rechner ist rein mechanisch aufgebaut; elektromechanische Relais ware noch nicht etabliert.

Tabellen wurden auch tabellarisch berechnet: In den ersten Spalten waren die Argumente aufgeführt; dann wurde zeilenweise Spalte für Spalte aus den Werten in den schon berechneten Spalten die nächste Spalte gefüllt. Daher erkannte Babbage die Notwendigkeit mehrerer Register für die Spaltenwerte, sowie einer freien, programmierbaren Kombination der Werte der nächsten Spalte als arithmetische Operation aus anderen Spalten. Es war normalerweise lediglich eine Iteration über die Spalten notwendig, weshalb auch Zuse bei der Z3 nur eine Gesamtschleife vorgesehen hat.

Da negative Zahlen als Absolutbetrag mit getrenntem Vorzeichen gehalten wurden, war ein erheblicher Aufwand für die Bildung des Übertrags bei Additionen und Subtraktionen zu leisten. Daher plante Babbage, alle Operationen zentral auszuführen (*mill*, d.h. Fabrik) und eine große Anzahl von Speicherregistern vorzusehen. Dies erlaubte es aus, Multiplikation und Division über feste (Mikro-) Programme als Hardware zu realisieren.

Da im mechanischen Bereich ein Rückwärtsdrehen der Speicherscheibe mit wenig Aufwand möglich ist, gab es für Babbage keinen zwingenden Grund, nach Alternativen zu suchen. Also ging er immer davon aus, dass ein Auslesen durch Zurückdrehen auf Null zu bewirken ist. Dies führte dazu, dass die ausgelesene Zahl, so sie denn weiter benötigt wurde, in einem zweiten Speicherelement dupliziert und danach wieder zurückgeschrieben wurde. Die Idee, die Zahl durch eine volle Umdrehung vorwärts zerstörungsfrei auszulesen, stand Babbage offenbar nicht zur Verfügung. Damit war der Aufwand für die Speicherstellen erheblich größer als in allen späteren Dezimalrechnern.

Ein iterativer Programmablauf in dem Sinne, dass die Iteration datenabhängig ist, ist für viele Anwendungen nicht notwendig. Am häufigsten war die Bestimmung von Funktionswerten durch Potenzreihen; hierbei kann vorab entschieden werden, wieviele Glieder der Reihe für eine gegebene Genauigkeit benötigt werden. Sämtliche durch Polynome, auch näherungsweise, berechenbaren Funktionen erforderten lediglich einfache lineare Operationen.

Die nächste Ebene der Komplexität wäre das Lösen von Gleichungssystemen, d.h. die

Inversion von Matrizen, für die datenabhängige Wiederholungen sinnvoll sind und von Babbage vorgesehen, aber erst spät detailliert wurden. Dies gilt insbesondere, wenn mit Koeffizienten nahe Null gerechnet werden muss, oder die Anzahl der Variablen so groß ist, dass eine analytische Lösung nicht sinnvoll ist. Ob Babbage iterative Lösungen zur Nullstellenbestimmung oder impliziter Funktionen konkret in Betracht gezogen hat, ist noch nicht geklärt.

In jedem Fall hatte Babbage geplant, Programmteile zu wiederholen, wenn oder bis ein Überlauf, insbesondere ein Übergang zu einer negativen Zahl, stattgefunden hatte. Nach bisherigem Kenntnisstand wollte er dazu die Karten um eine Anzahl vor- oder zurückbewegen; leider hat er diesen Teil nicht genau ausgeführt. Es wäre zu klären, ob dabei eine Anzahl von Karten angegeben wurde, und wenn ja, in welchem Zahlsystem (binär oder dezimal) die Anzahl gezählt werden sollte. Da er beim Dividieren und Multiplizieren Zähler eingesetzt hat, hatte er die Technik hierfür zur Verfügung.

Das von Ada Lovelace publizierte Beispielprogramm zur Berechnung der Bernoulli-Zahlen war im übrigen auf den damaligen Entwürfen nicht allgemein, d.h. automatisch für beliebig viele Register, durchführbar, da es keinen indizierten Speicher gab.

Eine *Analytical Engine* wurde nie gebaut, da Babbage stets an der Verbesserung des Entwurfs gearbeitet hat.

Der MERAC Evaluator bewegt die Ziffern stoßweise, während die AE die Ziffernscheibe für die notwendige Zeit an eine Welle gekoppelt hat. Da das Ziffernsignal im MERAC ohnehin ein Freigabesignal ist und keine Ziffernimpulse auf dem Datenbus gesendet werden, wäre dies eine gute Option gewesen, die allerdings mechanisch aufwändiger ist und daher (noch) nicht realisiert wurde, auch wenn die Zuverlässigkeit und Geschwindigkeit wesentlich besser wären.

4.2. ENIAC (Eckard und Mauchly, 1946)

Der ENIAC wurde als Entlastung für Berechnungen geplant, die damals in Rechensälen von Personen mit mathematischer Vorbildung mittels mechanischer Tischrechenmaschinen ausgeführt wurden. Letztere konnten den Wert eines einstellbaren Registers zu einem weiteren Register addieren oder subtrahieren. Zudem war für Multiplikationen und Divisionen ein Zählregister vorhanden. Es ist also nicht überraschend, dass der ENIAC aus zwanzig 10-stelligen Registern für Dezimalzahlen besteht, zu denen der Wert eines anderen Registers addiert (oder subtrahiert) werden konnte. Für Multiplikationen und Divisionen waren Zusatzeinrichtungen an einigen Registern vorgesehen, um die Programmsteuerung zu entlasten.

Als Ziffernspeicher wurden mit 10 Vakuumröhren (Doppeltrioden) aufgebaute Ringzähler (Schieberegister mit genau einem gesetzten Bit) verwendet; sie sind also das elektronische Analogon einer drehbaren Scheibe.

Ein wesentlicher Fortschritt in der Architektur war die Erkenntnis, dass zum Auslesen weder eine Rückwärtsbewegung notwendig ist noch die gespeicherte Zahl verloren gehen muss. Dazu wird der Ringzähler einmal zyklisch rotiert (10 Takte); und die Anzahl der Takte nach dem Übergang 9 -> 10 wird als Wert gesendet. Dies ist auch mechanisch kein Problem und hätte den Aufwand beim Speicher der AE mehr als halbiert.

Da das Schieberegister aus binären Elementen aufgebaut ist, kann es unmittelbar zu Beginn auf einen Anfangszustand zurückgesetzt werden. Anders als bei der *Analytical Engine* konnten also die Akkumulatoren unmittelbar überschrieben werden.

Beim MERAC ist es vorteilhaft, die Ziffern nicht als Einzelimpulse, sondern als Freigabesignale darzustellen; insbesondere wird dadurch der — im ENIAC nicht vorhandene — Modifikator einfacher.

Der ENIAC verwendet eine zusätzliche Taktzeit, um bei Subtraktionen eine Eins in der untersten Stelle zu addieren. Im MERAC wird dazu der ansonsten unbenutzte Einerübertrag in der niedrigsten Stelle verwendet.

Die Programmierung des ENIAC erfolgte durch Impulse an die Steuerungen der Akkumulatoren, die bei Ende der — mehrfach wiederholbaren — Operation neue Impulse erzeugten. Hierzu wurden — wie bei elektronischen Analogrechnern — Kabel gesteckt; es gab also ursprünglich noch nicht einmal Datenträger für Programme. (Die Tabellenspeicher wurden später dazu verwendet). Dieses Verfahren, obwohl Parallelarbeit erlaubend, wird weiterhin als unzweckmäßig und unübersichtlich angesehen.

Ob und inwieweit das Beispiel der Bernoulli-Zahlen auf dem ENIAC programmierbar war, wurde m.W. noch nicht untersucht.

4.3. Harwell Decatron Computer (1952)

Das Decatron ist eine dezimale Zählröhre mit Glimmstrecken, die 1950 angekündigt wurde. Es war damit deutlich günstiger als 10 Doppeltrioden, wenn auch langsamer. Damit wurde ein Rechner mit 20 (max. 90) Registern gebaut, der im Original im TNMOC (The National Museum Of Computing) noch vorgeführt wird.

Es werden wie beim ENIAC die Ziffern parallel, aber die einzelne Ziffer als Anzahl von Pulsen seriell übertragen; auch erfolgt das Auslesen durch zyklische Rotation mit 10 Impulsen. Jeder Befehl gab, neben dem Befehlscode in der ersten Ziffer, das Quell- und Ziel-Register als zwei Dezimalziffern an.

Wie bei der *Analytical Engine* war ein unmittelbares Überschreiben der Speicherstellen nicht möglich; es war daher notwendig, nicht mehr benutzte Speicherstellen beim letzten Auslesen zu löschen.

Da in den Rechenpfaden Relais verwendet wurden, obwohl die Decatrons um den Faktor 100 schneller waren, war die Maschine recht langsam, aber sehr zuverlässig (bis zu 10 Tagen unbeaufsichtigt). Sie war nicht schneller als ein geübter menschlicher Rechner mit einem mechanischen Tischrechner, nur dass letzterer nach einigen Stunden eine längere Pause brauchte und weniger zuverlässig war.

Die Programmierung erfolgte durch Lochstreifen; fünf aufeinanderfolgende Dezimalziffern ergaben einen Befehl. Durch Wechsel des Lochstreifenlesers waren einige wenige Unterprogramme möglich. Zwar konnten Befehle aus den Registern ausgeführt werden — es war also ein speicherprogrammierbarer Rechner — aber dies war nicht der Normalfall.

Die Maschine hat die größte Ähnlichkeit zum dezimalen MERAC.

4.4. Zuse Z3 (1941)

Die Z3 von Zuse war ein binärer 22-Bit Relaisrechner mit 64 Speicherworten, der bereits 1941 Gleitkommaarithmetik beherrschte, die erst 1954 mit der IBM 704 kommerziell verfügbar wurde.

Die Z3 ist eine Ein-Adress-Maschine mit einem Akkumulator und Hardware-Multiplikation und Division. Die Programmierung erfolgte über 8-Kanal Lochstreifen, wobei ein Befehl nur 8 Bit benötigte und daher das Lesen des Lochstreifens keine wesentliche Verzögerung bedeutete.

Programmverzweigungen und -Schleifen waren nicht vorgesehen; einzig konnte ein Lochstreifen zusammengeklebt werden und so eine Tabellenzeile aus Eingabewerten berechnen und auszugeben.

Erst die Z4 wurde später so umgebaut, dass sie bedingte Verzweigungen durchführen konnte. Dabei wurde das Programmband bis zu einer Marke bewegt.

4.5. Atanasoff-Berry-Computer (1942)

Der ABC war ein serieller Binärrechner, der als Speicher für die Daten rotierende Trommeln mit Kondensatoren verwendete, also dynamischen Speicher. Für jede Spur der Trommel gab

es ein Rechenwerk; es war also der erste Vektorrechner. Die Programmierung erfolgte über Lochkarten.

Er wurde zwar gebaut und getestet, aber wegen des Krieges nicht eingesetzt und vergessen, bis er beim Patentstreit um die ENIAC-Patente wieder bekannt wurde.

4.6. Pilot ACE (1950)

Als speicherprogrammierter serieller 32-Bit Binärrechner mit 1 MHz Bittakt ist die von Alan Turing 1946 spezifizierte Pilot-ACE deutlich anders als die anderen Rechner.

Sie verwendet ausschließlich Zwei-Adress-Befehle zu Transfer eines Speicherwortes an ein Zielregister, wobei einige Zielregister arithmetische (und andere) Operationen bewirkten, insbesondere eine Addition zum bisherigen Inhalt.

Der Datenbus zum Transfer ist expliziter Bestandteil des Konzepts. Jeder Befehl öffnet ein Tor, das die Bitfolge eines Registers auf den Bus gibt, und ein anderes Tor eines anderen Registers, das diese Bitfolge aufnimmt. Ein Modifikator, der die Bitfolge modifiziert, war jedoch nicht vorgesehen, dies wurde durch unterschiedliche Tore am Ein- oder Ausgang eines Registers erreicht.

Der Speicher umfasste 384 32-Bit Register (mittels Ultraschall-Verzögerungsleitungen) für Programm und Daten.

Wie beim ENIAC wurden Lochkarten als Ein- und Ausgabemedium, verwendet, die bei zeilenweiser Abtastung (80 Spalten in 12 Zeilen) 20 Zahlen pro Sekunde liefern konnten und damit wesentlich schneller als Lochstreifen (mit etwa 20 Ziffern pro Sekunde) waren, so dass Zwischenergebnisse auf Lochkarten ausgegeben werden konnten. Die Dezimal-Binär Konvertierung in beiden Richtungen konnte und wurde wegen der hohe Rechengeschwindigkeit während der Wartezeiten auf die nächste Zeile der Lochkarten erfolgen und erforderte nur wenige Befehle.

Als speicherprogrammierbarer Rechner konnte die ACE beliebig Unterprogramme definieren (nicht rekursiv).

4.7. Anmerkungen

Der ENIAC sei bewusst als Dezimalrechner entworfen worden, weil der Aufwand mit 280 Röhrenfunktionen pro Register bei 10 Dezimalstellen geringer sei als die geschätzten 450 Röhrenfunktionen für einen 30-Bit Akkumulator (*First Computers*, S. 130). Dies nimmt offenbar einen 30-Bit Paralleladdierer an, der jedoch dann etwa den Faktor 10 schneller gewesen wäre. Hingegen benötigt ein 30-Bit serieller Rechner mit seriellem Addierer weniger als 100 Röhrenfunktionen pro Register. Somit einem Geschwindigkeitsverlust des Faktors 3 ein Drittel des Aufwands entgegen; es wäre sinnvoller gewesen, drei serielle Rechner zu bauen.

Ein serieller 21-Bit Rechner, d.h. etwa 6 Dezimalstellen, benötigt nur 60 Röhrenfunktionen anstelle von 280 für 10 Stellen; d.h. ein Viertel des Aufwands wird durch halbierte Geschwindigkeit erkauft. Erlaubt man, wie beim ENIAC ohnehin vorgesehen, die Kopplung von zwei Registern zu einem 41-Bit Register, sollten auch Aufgaben, die mehr 6 Dezimalstellen erfordern, lösbar werden. Alle anderen profitieren von der verdoppelten Effizienz.

Da im ENIAC 20 Takte bei 100kHz Taktfrequenz verwendet wurden, sind das 200µs pro Addition oder 5000 Additionen pro Sekunde. Mit dem von Alan Turing für die ACE spezifizierten 1MHz Takt bei 32 Bit Wortlänge sind durch 32µs Takt etwa 30´000 Addition pro Sekunde. Mit 21 Bit ergeben sich 50´000 Additionen pro Sekunde, also die zehnfache Leistung des ENIAC.

Erstaunlicherweise sind keine Versuche bekannt, nach der Publikation von 1950 Ringzähler oder Schieberegister mit Glimmröhren einzusetzen. Glimmröhren kosteten 1966 weniger als

1 DM pro Stück, während eine Doppeltriode E80CC für Digitalbetrieb damals 16 DM kostete. Damit hätte der reine Speicher für einen seriellen Rechner mit 21 Bit pro Register weniger als 40 DM anstelle von mehr als 200 DM für ein Schieberegister mit 10 Doppeltrioden gekostet.

Vorbehaltlich einer genauen Untersuchung halte ich die Pilot-ACE, nachdem sie vier Jahre später als der ENIAC benutzt werden konnte, für den für etliche Jahre schnellsten Rechner der Welt — allerdings auch den mit der kompliziertesten Programmierung.

Entscheidend für die Architektur war und ist die jeweils verfügbare Speichertechnologie. So konnte die ACE durch die Verzögerungsleitungen mehr als die Akkumulatoren und damit Programmspeicher bereitstellen. Atanasoff-Berry konnten hatten durch den dynamischen Kondensatorspeicher mehrere Additionen parallel durchführen. Der HAC (Harwell) basierte auf dem Decatron als Speicherelement. Erst der Kernspeicher machte große Speicher ökonomisch; mit ihm begann die wirkliche Ära der speicherprogrammierten Computer.

5. Literatur

John Manley, Elery Buckley: ”Neon Ring Counter. Electronics, January 1950

(wird ergänzt)