

IAS, EDVAC und EDSAC: Von Neumanns Rechner und ihre Verwandten

Seminar: Geschichte der Rechnerarchitektur

Martin Fritz
TU München
mfritz@pt.lu

Eric Esch
TU München
ericesch1998@hotmail.de

ABSTRACT

In diesem wissenschaftlichen Artikel soll die Von-Neumann Architektur vorgestellt werden. Neben einer historischen Präsentation des Erfinders der Architektur, John von Neumann, sollen anhand von 3 konkreten Maschinen, der IAS-Maschine, der EDVAC und der EDSAC, die Bauart erklärt werden. Neben kurzen Einführungen zu den Computern werden unter anderen deren Funktionsweise verdeutlicht, Beispiele zu Befehlen gegeben und Vergleiche unter einander geführt.

Dieser Artikel soll also neben Dokumentationszwecken auch über die damalige Revolution der Computer berichten und erklären inwiefern diese die moderne Computerwelt beeinflusst hat und unser Verständnis über die Rechner für immer geprägt hat.

Keywords

Geschichte der Rechnerarchitektur, EDVAC, EDSAC, IAS-Computer, Von-Neumann Architektur, Harvard Architektur, John von Neumann

1. EINLEITUNG

Computer prägen seit jeher die physikalische Welt. Ob Supercomputer in der Wissenschaft, Laptops in der Arbeitswelt oder auch mobile Computer in unserem Alltag (man denke an Smartphones, Tablets und co.); alle haben etwas wichtiges gemeinsam: Sie werden immer schneller, effizienter und praktischer. Alle paar Jahre wird eine neue Architektur vorgestellt, die die moderne Welt verändert. Computer benutzen immer weniger Energie um mehr zu leisten und bringen dabei unsere Gesellschaft voran. Mit der kürzlich vorgestellten neuen Zen 2 Architektur von AMD, die Performancesteigerung für weniger Strom verspricht, stellt sich die Frage, wo Computer, wie wir sie heute kennen, ihren Ursprung finden. Nicht immer konnten nämlich komplexe mathematische Rechnungen und Probleme in Sekundenschnelle gelöst werden. Nach der Suche zum Ursprung der modernen Computer stößt man sehr schnell auf John Von Neumann und die revolutionäre Architektur, die er zu seinen Lebzeiten erfand und dokumentierte. Dieser wissenschaftliche Artikel befasst sich mit den Änderungen, die mit diesen Veränderungen der Computerbauarten einhergehen, inwiefern die Von-Neumann Architektur Vorteile vorweisen kann und wie die ersten Von-Neumann Rechner aussahen. Dabei wird sich mit dem IAS Computer, der EDVAC und EDSAC Maschinen näher auseinandergesetzt um die Architektur konkret vorweisen und untersuchen zu können.

2. JOHN VON NEUMANN

John (geb. Janós) Neumann wurde als ältester von drei Söhnen am 28. Dezember 1903 in Budapest geboren. Seine Familie war wohlhabend, denn sein Vater war Bankier und wurde auch später für seine Dienste zugunsten des Österreichisch-Ungarischen Reiches mit einem Adelstitel belohnt. Als Johns Name eingedeutscht wurde, wurde dem Namen deshalb ein "von" hinzugefügt, um diesen Adelstitel würdigen zu können. Bereits früh wurde das Talent des jungen Von Neumann erkannt. Nach einer Absprache mit seinem Vater wurde schnell Privatunterricht für den zukünftigen Wissenschaftler organisiert. Unter der Betreuung eines Professors und Tutoren wurde dem Jungen Mathematikunterricht gegeben. Nachdem er seine Matura 1921 abschloss, schrieb er sich an der University of Budapest ein, verbrachte jedoch die meiste Zeit an der ETH in Zürich und an der Universität in Berlin. Für seine Studien als besonders nützlich erwies sich der Fakt, dass er in seiner Kindheit Deutsch, Französisch, Englisch und Italienisch lernte. Sein Vater war nämlich der Meinung, dass es wichtig sei Sprachen zu lernen um in den anderen Teilen Europas erfolgreich zu werden. An der ETH schloss er nebenbei ein grundständiges Studium in Chemieingenieurwesen ab, kam aber weiterhin für Prüfungen nach Budapest zurück. Hier erlangte er, ohne die Kurse jemals zu besuchen, zeitgleich mit dem grundständigen Studium in Chemie, einen Dokortitel in Mathematik. Er arbeitete in dieser Zeit viel an wissenschaftlichen Artikeln und nahm 1927 schließlich die Stelle als Dozent an der Berliner Universität an. Während er hier unterrichtete erlangte er nebenbei immer mehr Beachtung durch seine Artikel über verschiedene Themengebiete der Mathematik. Nach verschiedenen Stellenwechseln an diversen Universitäten zog er in die Vereinigten Staaten von Amerika um.

Hier wurde er 1933, drei Jahre nachdem er seine Frau Marietta Kovesi heiratete, vom Institute for Advanced Study als Professor eingestellt. Leider endete seine Ehe mit Marietta 1938. Er blieb jedoch nicht lange geschieden, denn bereits im gleichen Jahr heiratete er seine zweite Frau Klara Dan. Mit Anbruch des zweiten Weltkrieges begann er sich an immer mehr Projekten außerhalb des Institutes, an dem er eingestellt war, zu beteiligen. So unterstützte er, wahrscheinlich aus Ablehnung des Nazi-Regimes, das amerikanische Militär in Aberdeen. Hier half er bei der Lösung von ballistischen Problemen die unter anderem die Laufbahnen von Raketen, Granaten und anderen Projektilen betrafen. Ebenfalls leistete er wichtige Beiträge zum Manhattan Projekt bei. Es handelt sich hierbei um das Projekt zur Entwick-



Figure 1: János Neumann

lung der ersten Atombombe.

Da die Forschung und Entwicklungen der militärischen Projekte immense mathematische Rechenleistung erforderten und nicht mehr per Hand gelöst werden konnten, stieg das Interesse an einer automatisierten, effizienten Lösung zu Berechnungen seitens der Regierung. Von Herman Goldstine, eine Schlüsselperson bei der Entwicklung der ENIAC Maschine, wurde von Neumann 1944 in Computer initiiert. In diesem Themengebiet fing er gleich an zu arbeiten und brachte, nach kurzer Zeit, wichtige Ideen hervor die die moderne Computergewelt ungemein prägen sollten. Unter anderem brachte er 1945 „First Draft of a Report on the EDVAC“ heraus. Dieser Artikel offenbarte bereits verhältnismäßig ausgereifte Strukturen eines modernen Computerkonzeptes. Diese Computergestaltung sollte später als „Von-Neumann Architektur“ als Grundstein der modernen Computer benutzt werden. Zusätzlich zum Artikel, leitete er ebenfalls die Erbauung der IAS-Maschine. 1955 wurden jedoch Zeichen einer schwerwiegenden Krankheit bei ihm diagnostiziert. Sein verschlechternder Zustand sollte immer mehr Einschränkungen in seiner Arbeit mit sich führen bis er schließlich am 8. Februar 1957 im Walter Reed Krankenhaus in Washington verstarb [1].

3. VON-NEUMANN ARCHITEKTUR

3.1 Beschreibung

Eine Von-Neumann Maschine besteht aus 4 Hauptkomponenten. (siehe Figure 2)

Dazu gehören der Speicher, die Recheneinheit, das Leitwerk und das Eingabe- sowie das Ausgabegerät. Die Von-Neumann Architektur ist dadurch ausgezeichnet, dass die Maschine nicht mehr spezifisch für ein bestimmtes Problem entwickelt werden muss. Dadurch, dass die Befehle und Daten zusammen sich im Speicher befinden, ist der Rechner flexibler in seiner Anwendung. Ein von-Neumann Computer hat Zugriff auf sowohl gespeicherte Information und Befehle und kann beliebig zwischen den Beiden springen. Außerdem besitzt der Computer die Fähigkeit den Speicher zu verändern.

Dies ermöglicht es Programme, die davor sehr viel technischen Aufwand und eine Veränderung der Maschine sowie dessen logischen Aufbau, erfordert hätten viel einfacher und verständlicher auszuführen. Man kann auch von einem „programmgesteuerten Universalrechner“ reden.[2] Damit diese Flexibilität gewährleistet wird, ist der Speicher in Einzelteilen fester Größe eingeteilt. Diese Zellen besitzen eine Adresse womit der spezifische Inhalt im Speicher adressierbar ist.

Programme werden sequenziell abgearbeitet: Befehle werden in einer bestimmten Reihenfolge nur einzeln ausgeführt. Zunächst wird ein Befehl aus dem Speicher von der Adresse geholt, die vom Befehlszähler angegeben wird. Danach wird der geholte Befehl ausgeführt. Für die meisten Befehle sind zusätzliche Daten bzw. Operanden notwendig. So kommt es zum Beispiel bei einer Additions Instruction, die $a+b$ berechnet, dass der Programmierer dem Programm mitteilen muss, wo oder was a und b sind. Verschiedene Architekturen erlauben die Angaben von Konstanten, andere eventuell nur die Angabe von Adressen an denen sich Daten befinden. Je nachdem variieren solche Restriktionen auch vom spezifischen Befehl der verwendet wird. Neben zwei Quellen für a und b wird ebenfalls ein Zielort benötigt. Bei der EDVAC Maschine beispielsweise, wie später noch einmal verdeutlicht wird, wird innerhalb des Befehls eine Zieladresse angegeben an der das Resultat gespeichert wird. Dies muss aber nicht immer der Fall sein; bei der 80386 Architektur wird, im Gegensatz zur EDVAC Maschine, die Quelladresse für den Operanden a ebenfalls als Zieladresse verwendet. Obschon die Befehle wie vorhin erwähnt sequenziell, das heißt nacheinander, ausgeführt werden, kann man vom Befehlsstrom abweichen, indem Sprünge durchgeführt werden. Es wird also ein Sprungbefehl benutzt, dem eine Zieladresse angefügt ist. Der Computer verändert daraufhin den Befehlszähler, sodass an der angegebenen Adresse dann mit dem Programm weitergemacht wird. Auch hier führt die Maschine dann die Arbeit sequenziell wieder fort [2].

Ein Problem der von-Neumann Architektur ist der sogenannte Von-Neumann-Flaschenhals. Mit dem Flaschenhals wird der langsame Zugriff, über mehrere Taktzyklen, auf den Speicher gemeint. Dadurch, dass der Speicherzugriff lange dauert werden die anderen Komponenten des Rechners eingeschränkt, da teilweise Befehle und Daten schneller verwertet werden als sie abgerufen werden können. Dieser langsame Speicherzugriff kommt vom Bussystem der Architektur. Es gibt einen Adressbus und einen Datenbus über denen jeweils pro Zyklus nur ein Datum traversieren kann [2].

Heutzutage ist die von-Neumann Architektur weiterhin relevant und findet Gebrauch. Aus Effizienzgründen werden Befehle aber nicht mehr bloß der Reihe nach ausgeführt, sondern parallel abgearbeitet. Hierfür werden unter anderem mehr als einen Prozessor benutzt um Anweisungen gleichzeitig ausführen zu können. Auch der von-Neumann Flaschenhals besitzt weiterhin seine Relevanz. Speicherzugriffe sind immer noch langsamer als Rechengeschwindigkeit. Schneller Speicher ist sehr teuer und kann deshalb nicht in großen Kapazitäten verbaut werden. Aus diesem Grund muss der Speicher in einer Speicherhierarchie aufgeteilt werden. Daten auf die oft zugegriffen wird und die eine hohe Priorität haben werden in schnellerem Speicher abgelegt als

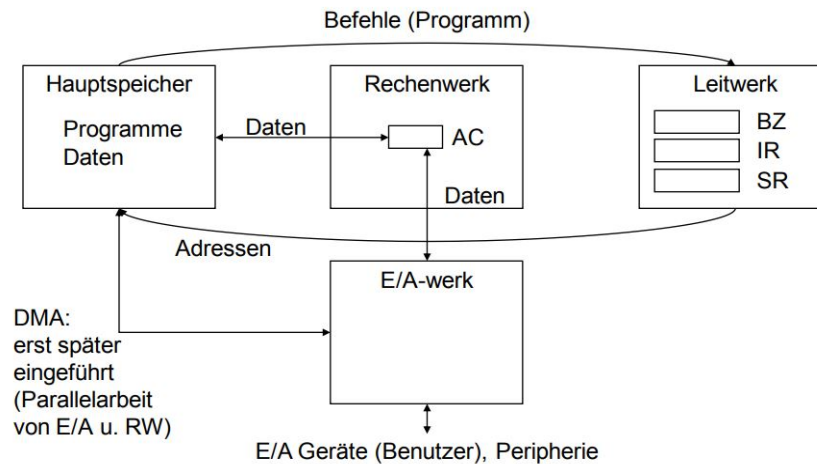


Figure 2: Von-Neumann Architektur

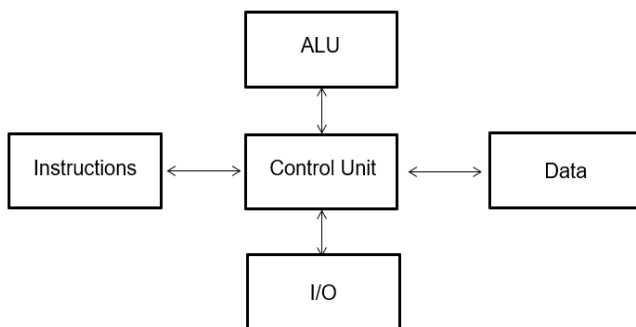


Figure 3: Harvard Architektur

Daten auf die der Nutzer nur gelegentlich Zugriff benötigt [2].

3.2 Vergleich mit der Harvard-Architektur

Eine Computerarchitektur die der Von-Neumann Architektur oft als konkurrierende Architektur gegenübergestellt wird, ist die Harvard Architektur. Die beiden Prinzipien haben gemeinsam, dass sie mehr oder weniger die gleichen Hauptkomponenten besitzen. Auch die Harvard Architektur sieht eine Kontrolleinheit, eine Recheneinheit (ALU) sowie eine Einheit zur Eingabe- bzw. Ausgabe vor.

Der fundamentale Unterschied ist, dass Computer, die nach dieser Art und Weise erbaut sind, nicht alle Programme und Daten in einer einzigen Speichereinheit wie bei der Von-Neumann Architektur halten. Anders als die Von-Neumann Bauart besitzen sie nämlich zwei Speichermedien. Eins davon wird für die Daten verwendet, im anderen Speicher werden die Programme bzw. Daten abgelegt. (siehe Figur 3) Was eventuell als Platzverschwendung bewertet werden könnte ist jedoch eine praktische Umgehung des Von-Neumann Flaschenhalses. Da die Recheneinheit über zwei Verbindung mit sowohl den Befehlen und den Daten gleichzeitig verbunden sind, ist sie nicht mehr durch den einzelnen Speicherzugriff beeinträchtigt. Tatsächlich können beide Speicher durch die Datenbusse parallel Daten an die CPU pro

Taktzyklus überbringen und ermöglichen somit eine Vermeidung des Problems der Von-Neumann Architektur.

Der Nachteil der Harvard-Architektur ist neben eines komplizierteren Systems ebenfalls eine teurere Hardware und einen höheren Bedarf an Platz (wegen der Präsenz von zwei benötigten Speichern) [3].

4. IAS

4.1 Einführung

Die IAS Maschine (Institute for Advanced Study: siehe Figur 4) war eine der ersten automatischen Maschinen, die mathematische Probleme lösen konnte. Das Projekt diese Maschine zu bauen, wurde von Von-Neumann selbst geleitet, gefolgt von seinem Team aus dem Institute for Advanced Study (Arthur Burks, Herman Goldstine, etc). Die Maschine verlief nach dem Prinzip der Von-Neumann Architektur. Also gab es einen Speicher aus einer Art Vakuum Tubes, einem Akkumulator welcher die Möglichkeit für Addition, Subtraktion und Shifting hatte und einem arithmetischen Register welche doppelte Wortlänge hatte um Berechnungen mit Multiplikation und Division zu halten. Die Wortlänge der Maschine war 40 Bit lang.[4]

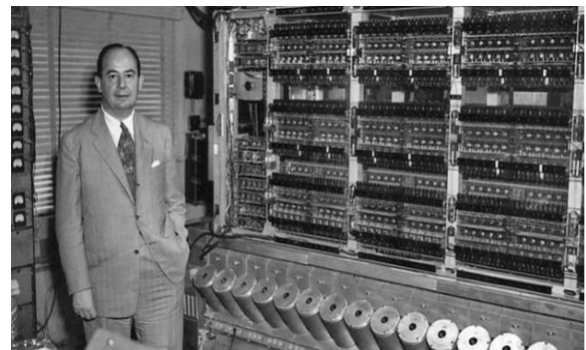


Figure 4: IAS, Institute for Advanced Study with John von Neumann

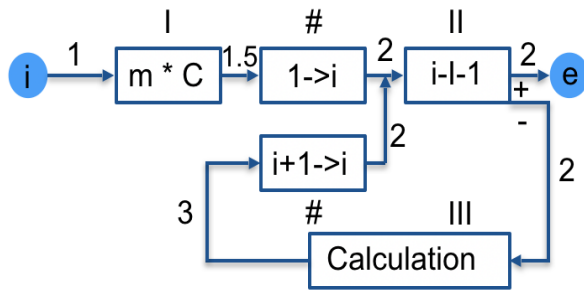


Figure 5: Flow diagram for IAS Machine

4.2 Programmentwicklung

Ein wichtiger Punkt dieser Maschine war, dass es nötig war ein mathematisches Problem das man hatte in Befehle übersetzen zu müssen. Dies geschah über ein Flussdiagramm, das von den Mathematikern gehandhabt wurde. In diesem Diagramm deuteten die Kreise auf Eingang/Ausgang, die Pfeile waren die Fließrichtung des Programms und die rechteckigen Boxen waren Berechnungsstellen, wobei man hier zwischen einer operational Box welche für eine normale Berechnung stand und einer mit zwei Ausgängen unterscheidete. Diese nannte man Alternative Box und repräsentierte, conditional branches (oft Schleifenausgänge) mit einem + Ausgang, wenn die Expression größer oder gleich 0 war und einem - Ausgang, wenn die Expression kleiner 0 war. Wenn eine Box mit einem Hashtag markiert war, hieß dies, dass es sich um eine substitution box handelte, welche Veränderungen der Variablen im Speicher bedeutete oder um eine assertion box was einer mathematischen Auswertung entsprach. Speicherstellen wurden mit großen Buchstaben und numerischen Suffixen benannt.[4]

Nachdem ein Diagramm fertig war, wurden am Ende die alternative und operational boxes mit römischen Ziffern nummeriert um somit einen Lauf für die benötigten Instruktionen zu haben. Erst nach all diesen Berechnungen war es möglich zu schätzen, wie viel Speicher das Programm benötigen würde. Somit konnte man dann die Speicherstellen zuteilen. Danach würde ein Ingenieur, die Daten ins binäre übersetzen und in den Speicher laden. Somit um ein Programm laufen zu lassen, gab es 4 Steps die man beachten musste. Man musste den Algorithmus mathematisch beschreiben, dann diesen Algorithmus in ein Flussdiagramm überführen (siehe Figur 5), dieses Flussdiagramm in eine Art Pseudocode überführen und diesen Code dann in den finalen Code umbauen indem nur noch die Befehle der Maschine benutzt werden und schlussendlich ins binäre übersetzen. Auf der Figur 5 sieht man nun ein ganz kleines Beispiel eines solchen Programms, wobei die Berechnung im dritten Punkt jetzt weggelassen wurde.[4]

4.3 Eingabe und Ausgabe der Maschine

Die IAS Maschine, war im Gegenzug zu anderen Maschinen etwas anders, denn sie besitzte keinen direkten Input oder Output. Man konnte also nichts eingeben noch ausgeben lassen. Das Prinzip dieser Maschine bestand darin, dass man Variablen und Konstanten per Hand in den Speicher eingab. Dazu benutzte man die Diagramme des Prob-

lems die ins binäre übersetzt wurden. Diese binäre Daten wurden dann in den Speicher geschrieben, also wurden sozusagen im Speicher Bitweise Schalter umgelegt um Daten zu schreiben. Diese wurden dann vom Programm benutzt und möglicherweise auch verändert. Nach dem Programmablauf musste man dann den Speicher per Hand inspizieren und dann die Daten aus dem Speicher lesen, indem man auf den Williams Tubes die Bits abgelesen hat. Danach wurden die Werte skaliert und man hatte das Resultat. [4]

4.4 Speicher der IAS Maschine

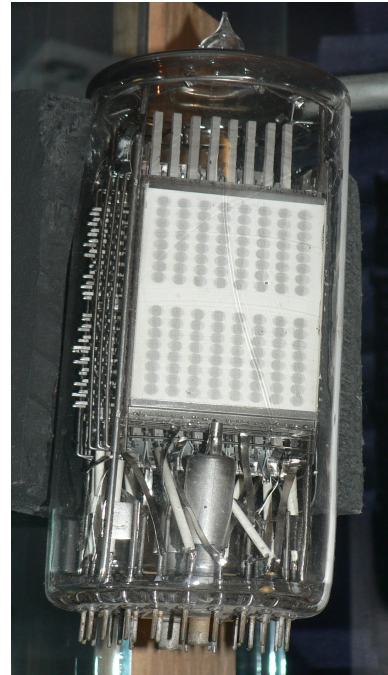


Figure 6: Selectron tube, Stanford museum of computing, William H. Gates building, Stanford University

Die IAS hatte Speicher für 2048 40-Bit Wörter, wobei Daten und Instruktionen wie von der Von-Neumann-Architektur beschrieben im gleichen Speicher lagen. Die Instruktionen waren 20 Bit lang und somit konnte man wie bei der EDSAC 2 Instruktionen in einem Speicherwort halten. Davon waren 8 Bit für den Opcode und 12 für die Adresse zuständig. Da aber nun 2 Instruktionen in einem Wort gespeichert werden können, ist es wichtig, dass man diese während der Nutzung unterscheiden konnte. Dies passierte durch benutzen eines ' in der Instruktion, somit würde z.B. Cc 30 die linke Instruktion aus dem Speicherplatz 30 benutzen und Cc' 30 die rechte davon. Diese Unterscheidung wird im Übergang zwischen Step 3 und Step 4 eines zu schreibenden Programms gemacht.[4]

Als die IAS Maschine gebaut/geplant wurde, dachte man daran den Speicher aus ungefähr 2300 Selectrons (siehe Figur 6) zu bauen. Diese waren eine Art Speicher, die nach dem Prinzip von Vakuum Röhren funktionierte. Jedoch war es schwieriger dies zu entwickeln als erwartet, was dazu führte, dass der Speicher nicht rechtzeitig fertig war. Somit wechselte man auf die Williams Tube (siehe Figur 7) welche nach



Figure 7: Williams-Kilburn tube from an IBM 701, 1951, at the Computer History Museum

dem gleichen Prinzip funktionierte aber einen Leuchtschirm besitzte, welcher zwischen 200 bis 2000 Bits gleichzeitig anzeigen konnte. Jedoch war die Williams Tube stark unzuverlässig und alterte ziemlich schnell, wobei Selectrons nachdem sie schlussendlich fertiggestellt waren viel zuverlässiger und schneller waren. Jedoch waren diese extrem teuer und wurden nie wirklich gebraucht. Schlussendlich besaß die IAS Maschine um die 1700 Williams Tubes.[4]

5. EDSAC

5.1 Einführung

5.1.1 Geschichte

Die EDSAC (Electronic Delay Storage Automatic Calculator: siehe Figur 9) ist heutzutage einer der wichtigen Meilensteine die die Rechnersysteme unserer Zeit prägen. Die Maschine wurde in der Cambridge Universität erbaut und verblieb bis zu ihrem Ende auch dort. Sie wurde von Maurice V. Wilkes und seinem Team aus Cambridge basierend auf John von Neumanns ersten Vortrages zur EDSAC erbaut. Dieser Vortrag wurde 1946 nach dem Krieg auf der Universität (Moore School of Electrical Engineering, University of Pennsylvania) gehalten und gab Wilkes Ideen, wie man die EDSAC konstruieren könnte. Somit begann die Konstruktion mit seinem Team im Jahr 1947 und die Maschine tätigte ihre erste Rechnung am 6.Mai 1949. Es ist wichtig zu erwähnen, dass die EDSAC von dem mathematischen Team von Cambridge konstruiert wurde und es ihnen nicht um Performance der Maschine ging sondern eher um die Zuverlässigkeit und die Möglichkeit Programme auszutesten, die davor nicht testbar waren. Also gab es im Großen und Ganzen viele Bearbeitungszeiten der Maschine und Modifikationen wurden auch fast gar nicht in Betracht gezogen. Dazu später jedoch mehr.[7]

5.1.2 Entscheidungen zum Aufbau

Die Maschine basiert wie vorhin erwähnt auf der Von-Neumann Architektur, so wie die EDVAC auch. Es blieb nur die Entscheidung ob man eine Festkommazahlen- oder Gleitkommazahlen Arithmetik benutzen wollte. Wobei jedoch schnell klar war, dass die Maschine eine Festkommazahl Maschine werden soll, da es eine große Anmutung an die Vakuum Röhre gewesen wäre diese so zu bedienen, obwohl die Fließkommaarithmetik bereits anerkannt wurde und Maschinen es bereits benutzten. Danach wurde die Maschine wie viele andere auch mit einer fixen Wortlänge und einem Akkumulator gebaut, diese Addition, Subtraktion und Multiplikation sowie shifting nach links und rechts ermöglichten. Außerdem bot die EDSAC einen festen Speicher an, der dazu benutzt werden konnte, Daten aus und in den Akkumulator zu laden, und geschriebene Programme oder Libraries abzusichern um sie später ohne erneutes schreiben benutzen zu können.[6]

5.2 Rechenwerk/Leitwerk

Um die Maschine so simpel wie möglich zu halten, entschied man sich für eine Einzeladdressform des Instruktionsformates/satzes. Dabei benutzte eine Instruktion 17 Bit (siehe Figur 10). Eine Instruktion bestand nun aus mehreren Teilen, wobei die ersten 5 Bits (Function Code) Funktionscode dazu dienten, welche Operation man benutzen wollte. Diese wurden außerdem als Buchstaben des Alphabets dargestellt und des Öfteren auch mit dem Buchstaben der zur Operation passte (z.B. A bei Addition). Anfangs gab es 18 verschiedene Operationen (siehe Figur 23), wobei später jedoch mehrere dazu gekommen sind. Anfangs gab es keine Zero-Test-Instruktionen und somit war man gezwungen das Vorzeichen des Akkumulators zu betrachten. Jedoch gab es die E-Instruktion die bei einem positiven Vorzeichen oder einem Nullwert im Akkumulator an die Adresse geht und diese Instruktion ausführt und die G-Instruktion die bei negativen Wert, das gleiche tat. Außerdem hatte die EDSAC keinen Divide Befehl. Jedoch konnte dies leicht durch Subroutinen nachgemacht werden. Das 7te bis zum 16ten Bit wurden als Adressen benutzt. Somit gab es 1024 mögliche Speicheradressen für Wörter, wobei anfangs nur 512 genutzt wurden und erst in 1952 auf 1024 aufgerüstet wurde. Das 6te Bit war ein anfangs ungenutztes Bit, das im Jahr 1955 jedoch benutzt wurde um Modifikationen von Adressen zu spezifizieren als das Index Register zur EDSAC hinzugefügt wurde. Das 17te Bit wurde dazu genutzt um anzugeben, ob man auf einem langen oder einem kurzen Wort im Speicher arbeiten möchte. Somit war die Wortlänge schlussendlich 35 Bit lang (mit einem Vorzeichenbit), jedoch wurde dies auf 36 hochgerechnet, da dies zuerst einmal ingenieurische Vorteile mit sich brachte aber auch, dass man zwei Instruktionen in einer Speicherzelle speichern konnte, in der man auch ein Wort speichern konnte. Somit war es schlussendlich möglich durch die Technik um auf Instruktionen der Länge von 17 Bit zuzugreifen auch auf Halbwörter zuzugreifen. Somit wurden in der EDSAC auch Wörter der Länge 16 Bit mit einem Vorzeichenbit (insgesamt 17 Bit) erlaubt. Somit wurde auch der Speicher mit Halbwörtern bezeichnet, um die Referenzierung auf eine Speicherstelle zu erleichtern und zu verschönern. Die durchschnittliche Arbeitszeit eines Befehls dauerte 1.5ms, für die Multiplikation 4.5ms und für die Division-Subroutine um die 200ms.[6]

Table 1. IAS computer instruction set.			
Inst #	Inst name	Abbrev	Description
1	$S(x) \rightarrow Ac +$	x	Copy number in Selectron location x into A.
2	$S(x) \rightarrow Ac -$	x -	Same as #1 but copy the negative of the number.
3	$S(x) \rightarrow AcM$	xM	Same as #1 but copy the absolute value.
4	$S(x) \rightarrow Ac-M$	x - M	Same as #1 but subtract the absolute value.
5	$S(x) \rightarrow Ah +$	xh	Add number in Selectron location x into A.
6	$S(x) \rightarrow Ah -$	xh -	Subtract number in Selectron location x from A.
7	$S(x) \rightarrow AhM$	xhM	Same as #6, but add absolute value.
8	$S(x) \rightarrow Ah-M$	xh - M	Same as #7, but subtract absolute value.
9	$S(x) \rightarrow R$	xR	Copy number in Selectron location x into R.
10	$R \rightarrow A$	A	Copy number in R to A.
11	$S(x) * R \rightarrow A$	x x	Multiply number in Selectron location x by the number in R. Place the left half of the result in A and the right half in R.
12	$A/S(x) \rightarrow R$	x ÷	Divide the number in A by the number in Selectron location x. Place the quotient in R and the remainder in A.
13	$Cu \rightarrow S(x)$	xC	Continue execution at the left-hand instruction at Selectron location x.
14	$Cu' \rightarrow S(x)$	xC'	Continue execution at the right-hand instruction at Selectron location x.
15	$Cc \rightarrow S(x)$	xCc	If the number in A is ≥ 0 , continue as in #13. Otherwise, continue normally.
16	$Cc' \rightarrow S(x)$	xCc'	If the number in A is ≥ 0 , continue as in #14. Otherwise, continue normally.
17	$At \rightarrow S(x)$	XS	Copy the number in A to Selectron location x.
18	$Ap \rightarrow S(x)$	xSp	Replace the right-hand 12 bits of the left-hand instruction at Selectron location x by the right-hand 12 bits of A.
19	$Ap' \rightarrow S(x)$	xSp'	Same as above, but modifies right-hand instruction.
20	R	R	Shift the number in A to the right 1 bit (left-most bit is copied).
21	L	L	Circularly left-shift the bits in A and R as an 80-bit quantity, leaving the most significant bit of A unchanged.

A: accumulator
 Cc: control conditional (conditional branch in modern parlance)
 Cu: control unconditional (unconditional branch or jump in modern parlance)
 R: arithmetic register
 S: Selectrons (memory in modern parlance)

Figure 8: IAS computer instruction set

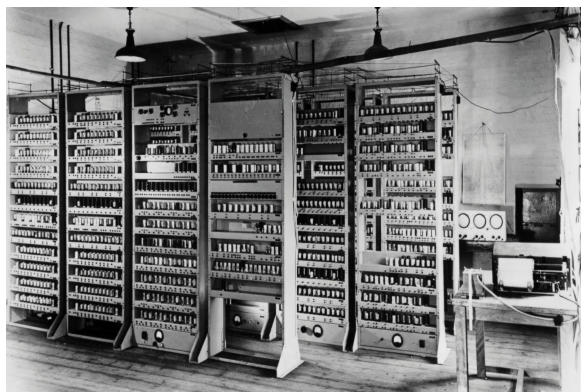


Figure 9: EDSAC I, June 1948 by Computer Laboratory, University of Cambridge

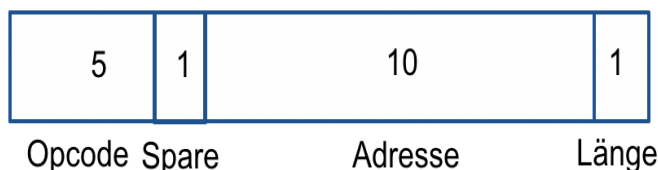


Figure 10: Instruction of the EDSAC

5.3 Input/Output der EDSAC

5.3.1 Aufbau

Um mit der Maschine kommunizieren zu können, benötigte man ein I/O-Werk, wie auch bei Von-Neumann-Architektur angedeutet. Somit wurde die EDSAC mit einem Fünf-spurigen Teleprinter Tape (siehe Figur 13) ausgestattet, welcher anfangs für Input und Output benutzt wurde. Die Maschine hing auch von der Geschwindigkeit des I/O ab, welche sich über die Zeit stetig verbessert haben.

Am Anfang wurde mit ungefähr 7 Zeichen pro Sekunde über einen elektromechanischen Tape-Reader eingelesen und über einen Teleprinter geschrieben, wobei jedoch der Input später mit bis zu 16 Zeichen pro Sekunde lesen konnte und im Jahr 1950 durch einen Fotoelektrischen Leser mit bis zu 50 Zeichen pro Sekunde. Somit wurde auch der ganze Code in einer Zeile untergebracht, damit dieser auf diese Streifen passte. Jedoch war es relativ schwierig und aufwendig diese zu managen und sauber zu lagern. Um nun zum Beispiel ein kleines "Hi" zu printen, würde ein Streifen mit folgendem Text gestanzt werden:

T64KGKZF05@06@07@ZF*FHFIFEZPF.[8]

5.3.2 5-Loch Tape

Das Inputtape besaß 5 Lochstellen pro Zeile, dies ermöglichte 32 verschiedene Kodierungsmöglichkeiten (siehe Figur 14. Jedoch wenn man das Alphabet und 10 Ziffern kodieren will, würde man bereits 36 Möglichkeiten kodieren müssen. Somit kam die Idee auf, einen Letter/Symbol-Shift einzuführen. Nun gab es also 2 Kodierungen die zwischen Symbolen und Buchstaben wechselte. Dieser Prozess ähnelt einem Interpreter von heute, welche die eingegeben Kodierungen entschlüsselt und ins binäre übersetzt. Ein weiteres interessantes Detail war, dass die Lochmaschinen entsprechend der Häu-

figkeit gebaut wurden. Dies bedeutet, dass eine Lochmaschine die häufigsten Buchstaben/Ziffern mit einer minimalen Anzahl an Bits kodierte, zum Beispiel wurde dann ein Buchstabe "e" mit nur einem Loch kodiert. Dies war ein wichtiges Detail, da die Locher sich nach einer Zeit abnutzen und somit nicht mehr sauber Löcher stanzen konnten. Somit konnte man weitere Wartungen der Maschinen vermeiden.[5]

5.4 Ausgeben des Wortes HI

5.4.1 Idee

Bei diesem kleinen Beispiel handelt es sich um ein Programm, das das Wort "HI" printen soll. Um dieses Programm nun auszugeben, muss man das Programm in den Speicher schreiben und das Programm ausführen und am Ende die Maschine resettet um das Ergebnis zu printen.

5.4.2 Funktionsweise

Das Programm würde so aussehen:
T64KGKZF05@06@07@ZF*FHFIFEZPF

Das Programm fängt mit dem Befehl T 64 K an, welcher den Ladepunkt an die Speicherstelle 64 setzt. Darauf folgt der Befehl G K welcher den Parameter @ auf den gerade gesetzten Ladepunkt setzt. Somit wurde dieser auf 64 gesetzt. Nun wird in jeder folgenden Instruktion die @ in der Adresse verwendet, die Zahl 64 drauf addiert. Dies ist ein wichtiger Punkt, welcher die Relocation des Programms ermöglicht, also konnte man das Programm in jegliche Speicherstellen laden. Es war oft der Fall, dass Programme an die Grenzen der Speichertanks geschrieben wurden, da dies es ermöglichte schnell das Programm auf dem Monitor zu kontrollieren ohne es in dem Bitmuster zu suchen. Nun startet jedoch erst das wesentliche Programm, welches mit einem Stopp Befehl Z F anfängt. Erst ab diesem Punkt war die Maschine gestoppt und man konnte den Speicher ablesen um sicherzustellen, dass das Programm sauber eingelesen wurde. Darauf folgten dann die Befehle O5@, O6@ und O7@, wobei der erste Befehl davon ein Letter Shift war und die anderen beiden die Buchstaben H und I printen. Darauf folgt erneut ein Stop Befehl Z F. Die folgenden Befehle sind zum überprüfen da und die letzten Zeichen E Z P F ermöglichen es zurück an den Programmpunkt 64 zu springen.[8]

5.5 Programme

Die EDSAC war eine vielfältige Maschine und konnte komplexe Programme lösen. Eines der ersten Programme wurde dazu benutzt Primzahlen zu finden, und tatsächlich gab es eine Zeitspanne, in der die EDSAC den Rekord für die größte gefundene Primzahl hielt. Aber nach kurzer Zeit fingen Interessierte an, kleine Spiele für die Maschine zu programmieren und so kam es dazu, dass die Monitore dazu benutzt wurden Tic-Tac-Toe darzustellen etc. Jedoch wurde die Maschine für jegliche Zwecke benutzt, z.B. in der Wirtschaft, Astronomie, X-Ray Kristallographie und Strukturen für Myoglobin, Atomare Berechnungen und mathematische Probleme wie Gleichungen.[10]

5.6 Programmieren in der EDSAC

Die EDSAC funktionierte wie oben erwähnt nach dem Prinzip, dass man Tape einlas, indem eine Codezeile eingelesen wird. Somit war die Programmation auf der Maschine

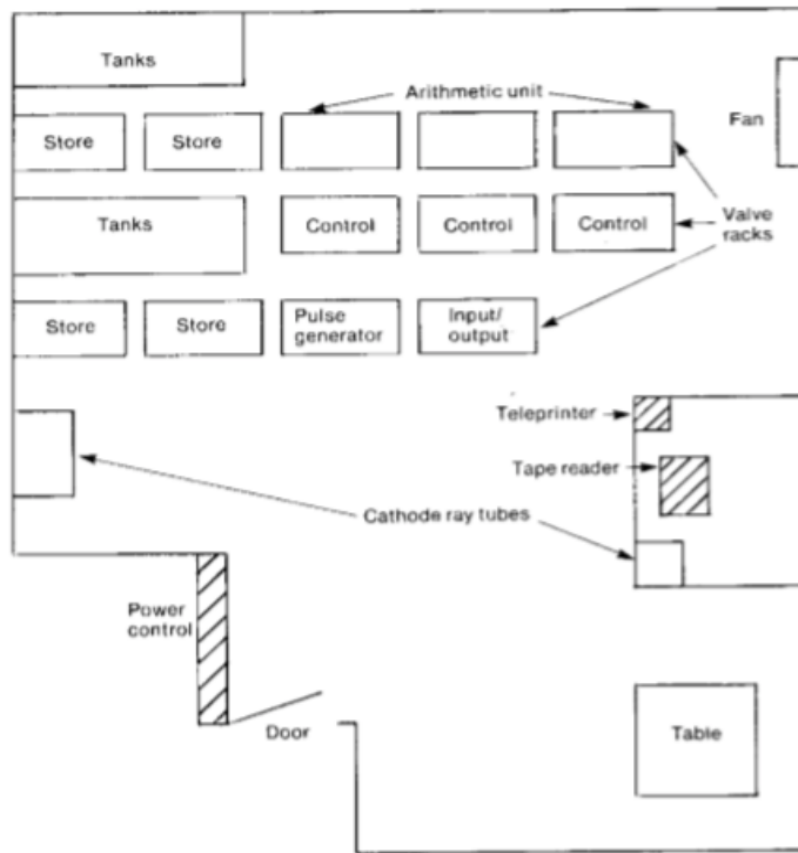


Figure 11: Room plan of the EDSAC

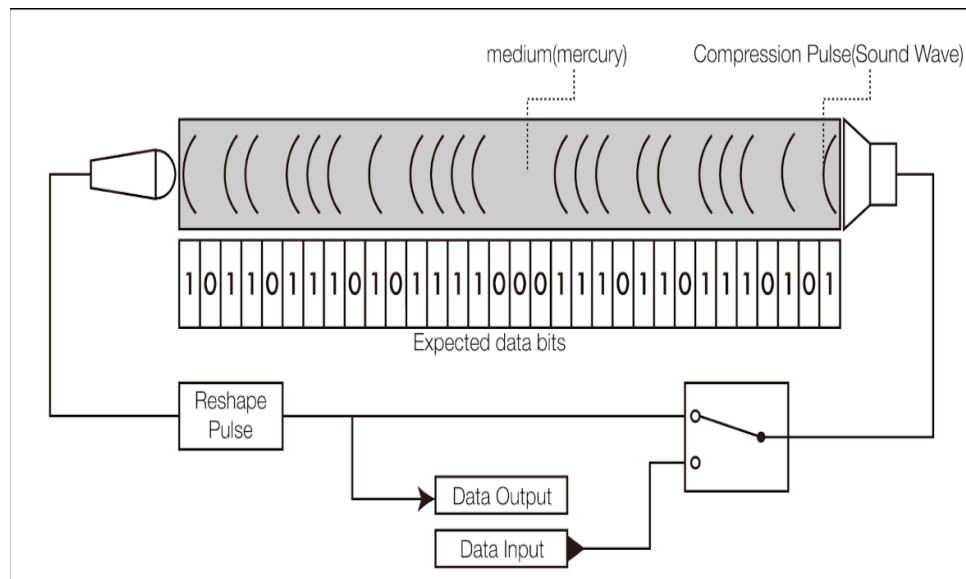


Figure 12: Signalflow inside Mercury Delay Line

sehr banal, da es kein Platz für Kommentare gab und es sich um eine reine Zeichenfolge handelte, die einfach abgelesen und ausgeführt wird. Somit wurden die Programm-streifen auch relativ kurz. Eine normale Instruktion war von dem

Format: Ein einziges Zeichen für die benutzte Funktion (z.B. A für Addition), einer Adresse in Dezimalformat und einem S oder L, das angibt, ob die Operation auf einer langen oder kurzen Zahl ausgeführt wird. Das Funktionszeichen wurde



Figure 13: EDSAC I, 1948, W. Renwick with 5 hole tape reader and Creed teleprinter.

nicht übersetzt sondern auf das Bit-Pattern des Teleprinters gemacht, wobei die dezimale Adresse ins binäre mit einer decimal-to-binary conversion Routine übersetzt wurde. Es ist außerdem interessant zu erwähnen, dass es nicht viel langsamer war in "normaler Sprache" zu schreiben, anstatt alles in binär zu kodieren. Somit waren EDSAC Programme und Sub-Routinen der Library immer in "normaler Sprache" gehalten. Es gab keine binäre Version außerhalb der Maschine, wie im Gegensatz zu heute, wo man jegliche Formen eines Programms gleichzeitig hat.[5]

5.7 Bauteile der EDSAC

Um nochmal auf zwei wichtige Bestandteile der EDSAC einzugehen, wird nun noch einmal der Akkumulator und die Mercury Delay Lines (Mercury Tanks) vorgestellt.

5.7.1 Akkumulator

Der Akkumulator war ein wichtiger Bestandteil der Maschine. Es war möglich Zahlen in den Akkumulator hinzu zu addieren oder zu subtrahieren, aber auch zwei Zahlen zu multiplizieren und das Produkt in den Akkumulator zu addieren. Außerdem besaß dieser 71 Bit und ermöglichte so volle Multiplikation von langen Zahlen. Wenn jedoch mit kurzen Zahlen gearbeitet wurde, wurden die linken Bits des Akkumulators benutzt. Es war ein Overflow im Akkumulator möglich, das wiederum den Programmieren mehr Freiraum gab um dies auszunutzen. Außerdem war shifting des Akkumulators möglich, also Multiplikation und Division mit 2 der Zahl im Akkumulator.

Die EDSAC besaß eine Art Alarm, welcher mit dem Akkumulator verbunden war. Dieser sprang immer an, wenn ein Überlauf des Akkumulators passierte und stoppte die Maschine. Jedoch war es von vielen Programmierern gewollt, den Akkumulator überlaufen zu lassen, weswegen es möglich gemacht wurde, diesen abzuschalten. Was oft dazu führte, dass vergessen wurde ihn wieder anzuschalten und bei anderen Programmen zu ungewollten Fehlern führte.[6]

5.7.2 Mercury Delay Lines

Der zweite extrem wichtige Bestandteil der EDSAC war der Speicher. (siehe Figur 15) Der Speicher war ein sehr schwierig implantierbarer Teil und nicht stark erforscht. Jedoch funktionierte die EDSAC wie auch die EDVAC mit Mercury Delay Lines. Die Speichergröße einer solchen Linie,

hing von der Länge der Tubes ab. Bei der EDSAC lag die Rohrlänge ungefähr bei 1.5m. Dies entsprach mehreren hundert Bits an Speicher pro Linie. Und 32 solcher Linien bildeten eine Mercury Tank. Danach kam Wilkes mit der Idee auf Batterien aus solchen Tanks zu bauen, und somit bildeten 16 Tanks eine Batterie, wobei die EDSAC zwei solcher besaß. Somit kam die EDSAC auf eine Speichergröße von etwa 1024 Wörtern von 35 Bits. Dies war eine schwierige Aufgabe, da extreme Genauigkeit beim Aufbau benötigt war und es keine thermale Deformation geben durfte, was wiederum nach einer Kühlung bedarf.[7.2] Eine solche Linie bestand aus einem langen hohlen Rohr, welches mit Quecksilber gefüllt wurde. An beide Enden der Röhre wurden Quarzkristalle angebracht die auf elektrische Impulse reagieren konnten. Das Prinzip bestand darin, dass wenn ein Impuls an den Input einer solchen Röhre kam, wurde vom Kristall ein Schallwelle abgestoßen, die dann im Quecksilber zur anderen Seite mit Schallgeschwindigkeit wanderte. Wenn nun ein Impuls durchlief und am anderen Ende auf den anderen Kristall traf, gab dieser wieder einen elektrischen Impuls nach außen hin weg. Nun wurde dieses Signal wieder mit einem Amplifier verstärkt und es gab die Möglichkeit das gesendete Signal an den Akkumulator zur Berechnung zu senden oder ihn wieder in den Kreislauf der Linie zu senden. Dieser Kreislauf des fortlaufenden Sendens des gleichen Impulses nannte man speichern. Dieser Speicher wurde dann von einem Receiver und Transmitter pro Tank gehandhabt, welche jedoch auf die Art und Weise gebaut wurden, dass man auch anderen Speicher verwenden konnte. Dies bedeutete, dass das Personal an Verbesserungen des Speichers gedacht haben. Wenn nun ein Wort von 35 Bit gespeichert wurde, das nur aus 1en bestand, wurden 35 Signale durch die Röhre gesendet, bei 0ern würde Stille herrschen. Wenn man jedoch ein Wort aus mehreren 1en und 0en hatte, wurde durch den Zeitabstand herausgefunden, ob nun eine 0 oder eine 1 gesendet wurde (siehe Figur 12).[9]

5.7.3 Probleme des Speichers

Der Speicher war eine innovative Erfindung, jedoch war der von der EDSAC benutzte Speicher nicht die beste Wahl. Dieser bestand nämlich aus Quecksilber, was ein extrem gesundheitsschädliches Material ist und am besten nur in kleinen Mengen verwendet werden sollte. Jedoch wurden hier Liter davon benutzt. Jedoch war es zu dem Zeitpunkt unvermeidbar, da man eine bestimmte Ausbreitungseigenschaft des Materials benötigte, welche für die Impulsübertragung nötig war. Wenn man heutzutage diesen Speicher erneut bauen wollen würde, würde das um die 150000 Euro kosten. Im Vergleich zu einer Festplatte die ein paar hundert Gramm wiegt und mehrere Terabyte speichern kann und knapp 50 Euro kostet ist das eine lächerliche Vorstellung um nur 1024 Wörter zu speichern. Es gibt jedoch in England ein Replikat der Maschine selbst, welche erst vor kurzem fertiggestellt wurde. Da es verboten war und auch zu teuer Mercury Delay Lines zu benutzen, wurde in diesem Fall Stahldraht benutzt. Dieser wurde um die eigene Achse verdreht. Das Prinzip ist das gleiche wie bei den Mercury Delay Lines, nur dass die Signale etwas spezieller übertragen werden.

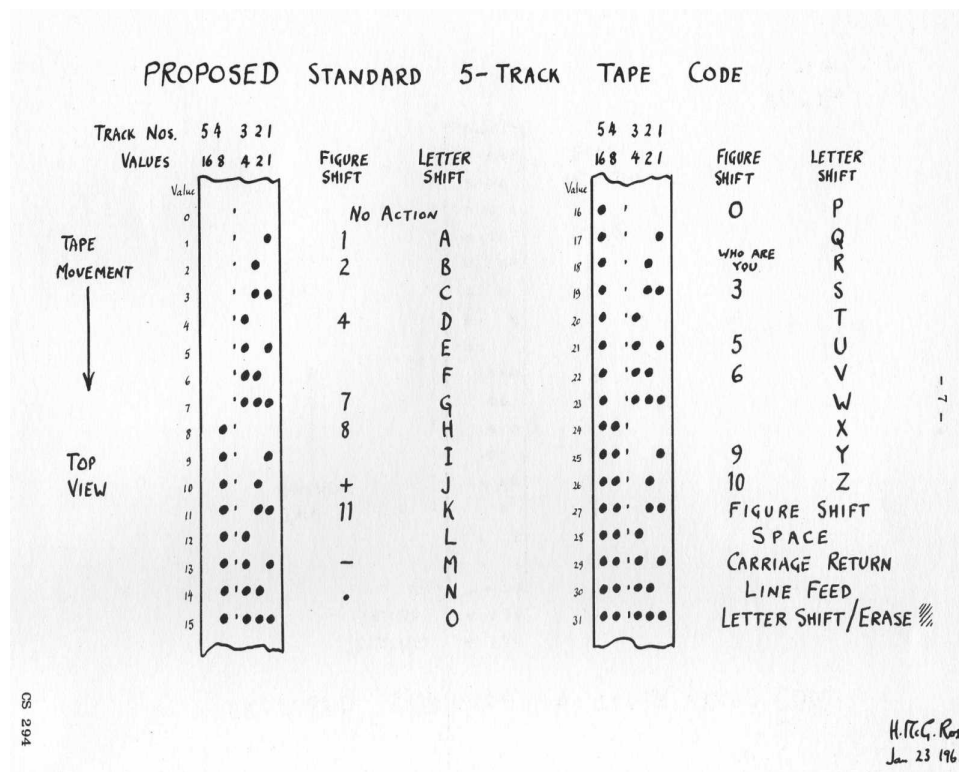


Figure 14: Five Hole Tape coding



Figure 15: Mercury Tank

6. EDVAC

6.1 Einführung

Die EDVAC (= Electronic Discrete Variable Automatic Computer) Maschine sollte, als Nachfolger der ENIAC, an dessen Erfolg anschließen und eine effizientere Lösung für die Berechnung mathematischer Probleme bieten. Da besonders das us-amerikanische Militär daran Interesse hatte, Projekte zu entwickeln, die den Rahmen der aktuellen automatischen Maschinen bzw. der verfügbaren menschlichen Fachkräften sprengen würden, finanzierten sie die Entwicklung der EDVAC und drückten diese ebenfalls voran. Allerdings sollte die Designphase und Konstruktion der Maschine nicht problemlos ablaufen wie sich noch zeigen wird [11].

6.2 Historischer Hintergrund

Um auf den Erfolg der ENIAC aufzubauen und eine effektivere, speicherprogrammierbare Maschine zu entwickeln schlossen sich von Neumann und ein Team der Moore School of Electrical Engineering zusammen. 1945 wurde kurzerhand „First Draft of a Report on the EDVAC“ von Johann von Neumann publiziert. Da allerdings nur er als Autor erwähnt wurde, kam es zu Streitigkeiten und späterhin zu Austritten innerhalb des Moore School Teams. Dies kommt daher, dass sich viele in ihrem Beitrag missachtet gefühlt hatten, weil die Öffentlichkeit nur Von Neumann mit dem Projekt in Verbindung setzte. Die Entwurfsphase wurde daraufhin bis auf weiteres auf Eis gelegt. [11]

Nach dem zweiten Weltkrieg wurde Oktober 1946 ein Treffen im Ballistic Research Labor in Aberdeen veranstaltet.



Figure 16: EDVAC, Electronic Discrete Variable Automatic Computer

Hier wurden unter anderem auch von Von Neumann 3 verschiedene Konzepte für eine neue potentielle EDVAC Maschine elaboriert [11].

Die EDVAC 1 war ein serieller binärer Computer, mit einem Speicher von 1000 Words, der die Addition, Subtraktion sowie Multiplikation durchführen konnte. Leider musste die Division noch programmiert werden und die Operationen wurden nicht intern überprüft. EDVAC 2 war ein dezimaler Computer, der die vier mathematischen Grundoperationen mit Festkommazahlen ausführen konnte. Hier war der Speicher gleich groß wie bei der EDVAC 1, aber die Operationen wurden intern geprüft. EDVAC 3 war ähnlich wie die EDVAC 2 jedoch mit einer Gleitkommaeinheit zur Berechnung von Gleitkommaoperationen ausgestattet. Der Speicher war außerdem 4-mal so groß wie bei den EDVAC 1 und EDVAC 2 Maschinen. Von diesen drei Konzepten wurde schließlich die EDVAC 1 übernommen und leicht weiterentwickelt. Heraus kam die EDVAC 1.5 Maschine mit Hardware Einheiten zur Division und einer Überprüfung der arithmetischen Operationen. (siehe Figure 17) Später wurde das Konzept der EDVAC 1.5 erweitert: Die EDVAC 1.5b bekam ein größeres Instruction Set. Ausserdem sollte das Gleitkommasystem für die mathematischen Grundoperationen nicht über Hardware sondern über Software gelöst werden, da es sonst zu Komplikationen bei der Zusammenstellung kommen würde.

May 1947 war das Design der Maschine endlich fertig und die Produktion konnte in der Moore School beginnen. Nach zwei Jahren wurde die EDVAC dann zur Zusammenstellung und zum Testen zum Aberdeen Proving Ground gebracht. Jedoch sollte das erste Programm erst 2 Jahre später, am 28. Oktober 1951, ausführbar sein. Mithilfe der EDVAC sollte eine symmetrische Matrix diagonalisiert werden. Erst ein Jahr später, ab März 1952, wurde die Maschine als voll funktionstüchtig angesehen und war täglich für 60-70% der Zeit zum Gebrauch zur Verfügung.

Die EDVAC wurde zum Beispiel zur Berechnung von Lauf-

bahnen der Sonne und des Mondes und der Messung und Auswertung von Satellitendaten benutzt.

Schließlich wurde 1962 beschlossen, dass die mittlerweile überholte EDVAC Ende Januar 1963 abgeschaltet werden sollte. Dies lag daran, dass nach langen 10 Jahren Betriebszeit, bereits weitaus schnellere, effizientere Maschinen zum Beispiel von IBM hergestellt wurden. Die mittlerweile überholte EDVAC konnte nicht mehr mit ihrer Leistung überzeugen.

Da die Maschine nach einem Ausschalten über die Weihnachtsferien nicht mehr funktionierte wurde das geplante Ende vorgezogen und somit war sie offiziell bereits Anfangs Januar außer Betrieb [11].

6.3 Befehle

Die EDVAC operierte mit Words die 44-Bit lang waren. Es gab 12 Instruktionen, wofür also mindestens 4 der 44 Bits belegt werden mussten. Die Instruktionen waren nach folgendem Schema aufgebaut (Figur 18):

OpCode QuelleA QuelleB Ziel NächsterBefehl

Für eine Liste der Instruktionen, siehe Figur 19.

Besonders hervorzuheben ist, dass es jeweils zwei Multiplikationsbefehle und zwei Divisionsbefehle gab. Wenn die Befehle mit Großbuchstaben benutzt wurden (Befehle 3 und 5 der Figur 19), wurden die zwei 44-bit Zahlen A und B aus den Quellen a und b entnommen und miteinander multipliziert oder dividiert. Die resultierende 86-bit Zahl wurde anschließend zu einer 44-bit Zahl abgerundet und an der Zieladresse g abgespeichert. Die Rechenbefehle mit Kleinbuchstaben (Befehle 4 und 6 der Figur 19) werden benutzt um präzisere Resultate abzuspeichern. Die Zahlen A und B werden aus den Quellen a und b entnommen und miteinander multipliziert oder dividiert. Anstatt aber das 86-bit Resultat abzurunden wird es in zwei 43-bit Zahlen aufgeteilt und an den Adressen g und g+1 abgespeichert. Somit kann der Programmierer auf ein viel größeres Resultat zugreifen, falls die Berechnungen größere präzisere Daten brauchen.

Neben der vielseitigen arithmetischen Möglichkeiten, die von der EDVAC geboten werden, ist besonders noch der "Extract" Befehl (Befehl 10 der Figur 19) wichtig vorzustellen. Dieser Befehl gibt die Möglichkeit, in einer ersten Phase, ein ausgewähltes Word nach links oder rechts zu shiften. In einer zweiten Phase können bestimmte ausgewählte Bits eines Words mit den Bits eines anderen Words ersetzt werden. Dieser Extract Befehl ist besonders für Adressmodifikationen der Befehle innerhalb der EDVAC benutzt worden [11].

6.4 Bauteile

6.4.1 Kontrolleinheit

Die Kontrolleinheit war bei der EDVAC für die unumgänglichen Computerfunktionen zuständig. So konnte man mithilfe dieses Bauteiles die Maschine ein- sowie ausschalten und auch Rechnungen anfordern und abbrechen. Eine weitere Funktion des Leitwerkes war es, den Modus Operandi auszuwählen. Es gab 4 verschiedene Modi zwischen denen man auswählen konnte um die Betriebsart der Maschine beeinflussen zu können:

- Normale Operation
- Ausführung der Befehle bis

	System	Interne Überprüfung	Speicher	Operationen
EDVAC 1	binär	-	1000 Words	Add, Sub, Mul
EDVAC 2	dezimal	Ja	1000 Words	Add, Sub, Mul, Div (+Festkomma)
EDVAC 3	dezimal	Ja	4000 Words	Add, Sub, Mul, Div (+ Gleitkomma)
EDVAC 1.5	binär	Ja	1000 Words	Add, Sub, Mul, Div

Figure 17: Zusammenfassung der verschiedenen EDVAC Konzepte

	OpCode	QuelleA	QuelleB	Ziel	NächsterBefehl
Anzahl an Bits	4	10	10	10	10

Figure 18: Format der EDVAC Befehle

1	A	a	b	g	d	Additionsbefehl
2	S	a	b	g	d	Substraktionsbefehl
3	M	a	b	g	d	Multiplikationsbefehl mit Abrundung
4	m	a	b	g	d	Genauer Multiplikationsbefehl
5	D	a	b	g	d	Divisionsbefehl mit Abrundung
6	d	a	b	g	d	Genauer Divisionsbefehl
7	C	a	b	g	d	Vergleichen von a mit b; Sprung nach d oder g je nach Resultat
8	W	a	b	g	d	Lesen bzw. Schreiben von Daten mithilfe eines Eingabe-/Ausgabegerätes
9	R	a	b	g	d	Lesen von Daten von Lochstreifen
10	E	a	b	g	d	Shiften von Bits eines Words nach Links oder Rechts; Überschreibung einiger Bits mit Bits von anderem Word
11	MR	a	b	g	d	Manueller Lesebefehl der die Bitschalter der Kontrolleinheit einliest und abspeichert
12	H	a	b	g	d	Stopbefehl

Figure 19: Liste der Befehle der EDVAC

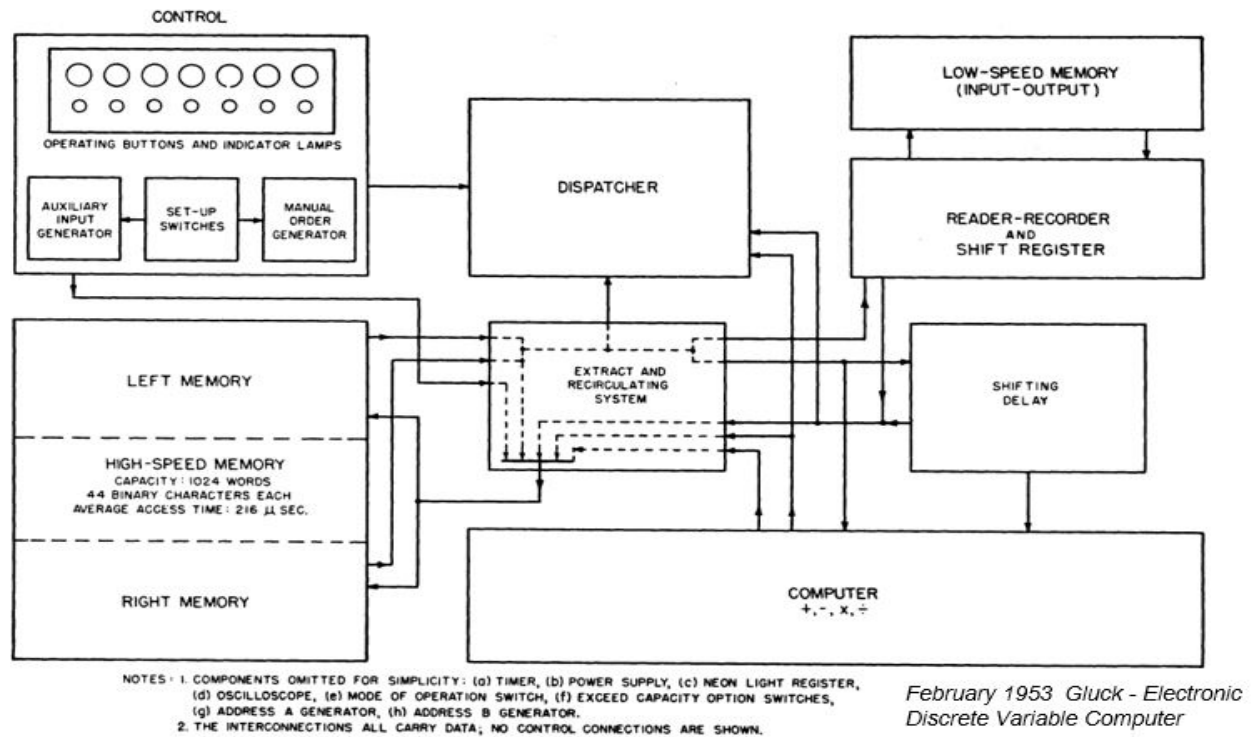


Figure 20: Schema der EDVAC Maschine

- Ausführung eines einzelnen Befehls
- Ausführung der Befehle die von den "special order" Schaltern bestimmt werden

Außerdem gab es 10 Neon Lämpchen, die den sogenannten "initial address register" darstellten. Heutzutage wäre dies der Inhalt des Befehlszählers. Unter diesen 10 Lämpchen befanden sich weitere 44 Lichter die den Inhalt ausgewählter Words anzeigen konnten. Ein Lämpchen stellte dabei ein einzelnes Bit dar, da ein Word in der EDVAC aus 44 Bits bestand [13].

6.4.2 High-Speed Memory

Die Daten wurden in Quecksilber Verzögerungsleitungen gespeichert. (siehe Figure 21)
Die High-Speed Memory Einheit bot Platz für 1024 Words. Näheres zu den "Mercury delay lines" wurde unter (5.7.2) aufgeführt [13].

6.4.3 Reader/Recorder und low-speed Memory

Von der Reader/Recorder Einheit wurde ein Schieberegister verwendet um asynchrone Daten zwischen dem langsamen Speicher und dem synchronen schnellen Speicher auszutauschen. Anzumerken gilt, dass der langsame Speicher auch zur Eingabe und Ausgabe verwendet werden konnte. Der langsame Speicher konnte außerdem verschiedene Formen annehmen. So gilt, dass jeglicher nichtflüchtige Speicher, der Informationen dauerhaft speichern konnte, unter Zuhilfenahme des Stroms der Maschine, gültig war. Beispiele wären Lochkarten sowie auch Fotofilme. Neue Ausrüstung konnte mit minimalem Aufwand hinzugefügt werden. Natürlich variierte die Leis-

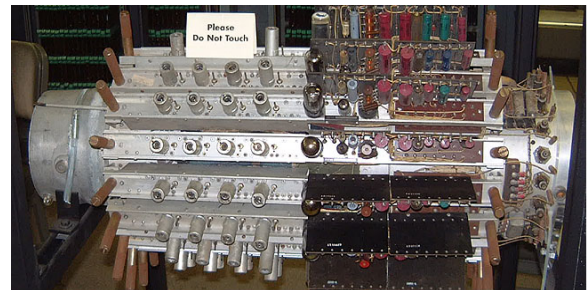


Figure 21: High-Speed Memory: Mercury Delay Lines

tung des Speichers und dementsprechend auch der Maschine je nach verwendetem Medium [13].

6.4.4 Dispatcher

Die Aufgabe des Dispatchers kann in vier Etappen zusammengefasst werden:

1. Erhalten von Bits.
2. Speichern der erhaltenen Bits.
3. Übersetzen der gespeicherten Bits zu einem Befehlsword.
4. Übergabe der Befehlsörter (in der richtigen Reihenfolge) an die zuständigen Einheiten der EDVAC zur Ausführung.

[13]

6.4.5 Shifting Delay

Für das Shifting Delay spielt der unter (6.3) aufgeführte "Extract" Befehl eine wichtige Rolle. Hier wird der erste Teil der Operation ausgeführt, nämlich das Shiften nach links bzw. rechts der Words [13].

6.4.6 Extract and Recirculating System

Beim Extract and Recirculating System handelt es sich wie beim Shifting Delay um ein Bauteil, dass für den Extract Befehl notwendig ist. Dieses Stück repräsentiert ein System von Bussen und Schaltkreisen, das von jeder Zahl und jedem Befehl durchlaufen werden muss. Hier wird der zweite Teil des "Extract" Befehls, sprich das Austauschen bestimmter Bits eines Words mit ausgewählten Bits eines anderen Words, ausgeführt [13].

6.4.7 Computer

Die Recheneinheit ermöglichte die 4 arithmetischen Grundoperationen. Für die Multiplikation und Division gab es jeweils zwei Befehle. Man konnte entweder die Operation ausführen und das Resultat abrunden oder das 86-bit Resultat aufteilen und in zwei Speicherzellen abspeichern. Die Rechnungen wurden gleichzeitig zweimal ausgeführt; bei Abweichungen wurde die Maschine gestoppt und die Kontrolleinheit benachrichtigt [13].

6.4.8 Timer

Der Timer ist dazu da um Zeitimpulse zu versenden, die zur Synchronisation der Maschine gebraucht werden [13].

6.5 Vergleich mit der von-Neumann Architektur

Wie unschwer zu erkennen ist, handelt es sich bei der EDVAC um ein bereits ausgesprochen ausgereiftes Beispiel der Von-Neumann Architektur. Obschon es eines der ersten Computer ist, die man der Architektur zurechnen kann, sind die Hauptmerkmale erkennbar. So besitzt man ein Leitwerk (-> Kontrolleinheit), ein Rechenwerk (-> Computer Einheit), ein Ein-/Ausgabewerk (-> low-speed Memory) und schließlich noch einen Hauptspeicher (-> high-Speed Memory) [2][13].

Einzig das Shifting Delay, das Extract and Recirculating System und die Dispatcher Einheit sind nicht mehr im Von-Neumann Schema (Figur 2) in dieser Form vorzufinden. Dies liegt daran, dass heutzutage meistens die Aufgaben dieser Hardwareeinheiten oftmals softwareseitig, unter anderem vom Betriebssystem, beseitigt werden [2].

6.6 EDSAC, EDVAC und IAS im Vergleich

	Addition	Multiplikation	Division	Wortlänge
EDSAC	1.50	4.50	200	35 Bit
EDVAC	0.90	2.90	2.90	44 Bit
IAS	0.06	0.71	-	40 Bit

Figure 22: Leistungen der Maschinen im Vergleich (Zeitmessungen in Millisekunden)

Der Figur 22 kann man entnehmen, dass die IAS außerordentlich viel schneller als die anderen beiden Maschinen ist. Die EDVAC ist etwas schneller als die EDSAC gewesen. Die Addition und Multiplikation werden fast zweimal schneller auf der EDVAC ausgeführt. Besonders die Division ist sehr langsam auf der EDSAC im Vergleich zur EDVAC. Obschon die IAS deutlich schneller als die anderen Computer ist, sollte man trotzdem den gravierenden Nachteil nicht ausser Acht lassen. Da es kein richtiges Input bzw. Output auf diesem Computer gab, sondern bloss manuelles inspizieren und schreiben des Speichers, war besonders für kurze Rechnungen der Zeitaufwand für die IAS grösser als für die EDVAC beispielsweise.

Anzumerken gilt es ebenfalls noch, dass die Maschinen im Vergleich zu heutzutage noch keine standardisierte Wortlänge hatten. In modernen Computer die ebenfalls (teilweise) auf die Von-Neumann Architektur basieren, findet man hauptsächlich 32-Bit oder 64-Bit Architekturen vor. Bei der Erfindung der aufgeführten Maschinen war noch kein Standard etabliert; daher kommt also der Unterschied bei den Wortlängen zwischen den Drei.

Abschließend kann man sagen, dass die drei Maschinen nicht viel anders sind als heutige Computer. Wie bereits in der Einleitung erwähnt ist die Leistung und Effizienz der Computer ein Bereich in dem immer wieder Neuerungen und Steigerungen eingeführt werden. Bereits in den 50-er Jahren konnte man aber Computer beobachten die sehr ähnlich zu den heutigen Computer waren. Das Prinzip des gemeinsamen Speicher für Programme und Daten hat sich über die Jahre hin bewährt und wird immer eine wichtige Revolution in der Geschichte der Informatik bleiben.

7. REFERENCES

7.1 Text

[1] D. Szász: John von Neumann, the Mathematician, The Mathematical Intelligencer (Volume: 33, Issue: 2, pp 42–51, 2011) [Online]. Available: <https://link.springer.com/article/10.1007%2Fs00283-011-9223-6> [Accessed: June 14, 2019].

[2] J. Weidendorfer : IN0004. Vorlesung, Topic: "Einführung in die Rechnerarchitektur" Technische Universität München, München, Wintersemester 2017/2018.

[3] A. Baba : Short Presentation, Topic: "Von-Neumann Architecture Vs Harvard Architecture" University of Turkish Aeronautical Association, 2019. Available: https://www.researchgate.net/publication/331160423_Von-Neumann_Architecture_Vs_Harvard_Architecture [Accessed: June 14, 2019].

[4] Dale Skrien: Debugging on the Shoulders of Giants: Von Neumann's Programs 65 Years Later Computer (Volume: 45 , Issue: 11 , Nov. 2012) [Online]. Available: <https://ieeexplore.ieee.org/document/6152078> [Accessed: June 14, 2019].

[5] MARTIN CAMPBELL-KELLY: Programming the EDSAC: Early Programming Activity at the University of Cambridge IEEE Annals of the History of Computing (Volume: 20, Issue: 4, Oct-Dec 1998) [Online]. Available:

TABLE 1
THE EDSAC ORDER CODE (1949)

A n	Add the number in storage location n into the accumulator
S n	Subtract the number in storage location n from the accumulator
H n	Copy the number in storage location n into the multiplier register
V n	Multiply the number in storage location n by the number in the multiplier register and add the product into the accumulator
N n	Multiply the number in storage location n by the number in the multiplier register and subtract the product from the accumulator
T n	Transfer the contents of the accumulator to storage location n and clear the accumulator
U n	Transfer the contents of the accumulator to storage location n and do not clear the accumulator
C n	Collate [logical <i>and</i>] the number in storage location n with the number in the multiplier register and add the result into the accumulator
R 2^{n-2}	Shift the number in the accumulator n places to the right
L 2^{n-2}	Shift the number in the accumulator n places to the left
E n	If the number in the accumulator is greater than or equal to zero, execute next the order that stands in storage location n; otherwise proceed serially
G n	if the number in the accumulator is less than zero, execute next the order that stands in location n; otherwise proceed serially
I n	Read the next row of holes on the tape and place the resulting five bits in the least significant places of storage location n
O n	Print the character represented by the five most significant bits in storage location n
F n	Check the last character output
X	Ineffective (no operation)
Y	Round the number in the accumulator to 34 bits
Z	Stop the machine and ring the warning bell

Figure 23: EDSAC OPCode Table

<https://ieeexplore.ieee.org/document/728229> [Accessed: June. 14, 2019].

[6] M.V. Wilkes, Olivetti and Oracle Res. Lab., Cambridge, UK: Arithmetic on the EDSAC IEEE Annals of the History of Computing (Volume: 19, Issue: 1, Jan-Mar 1997) [Online]. Available: <https://ieeexplore.ieee.org/document/560726> [Accessed: June 14, 2019].

[7] M.V. Wilkes: Early computer developments at Cambridge: The EDSAC Radio and Electronic Engineer (Volume: 45, Issue: 7, July 1975) [Online]. Available: <https://ieeexplore.ieee.org/document/5268249> [Accessed: June 14, 2019].

[8] MARTIN CAMPBELL-KELLY, "Tutorial Guide to the EDSAC Simulator" <https://www.dcs.warwick.ac.uk/edsac/> [Online]. Available: <https://www.dcs.warwick.ac.uk/edsac/Software/EdsacTG.pdf>. [Accessed: June 14, 2019].

[9] Matsuura Tomoya, "Art Project" https://matsuuratomoya.com/en/works/post-past_sotsuten/ [Online]. Available: https://matsuuratomoya.com/en/works/post-past_sotsuten/ [Accessed: June 14, 2019].

[10] J.M. Wheeler: Applications of the EDSAC IEEE Annals of the History of Computing (Volume: 14, Issue: 4, 1992) [Online]. Available: <https://ieeexplore.ieee.org/document/194052> [Accessed: June 14, 2019].

[11] M.R. Williams: The origins, uses, and fate of the ED-VAC, IEEE Annals of the History of Computing (Volume: 15, Issue: 1, 1993) [Online]. Available: <https://ieeexplore.ieee.org/document/194089> [Accessed: June 14, 2019].

[12] S.E. Gluck : The Electronic Discrete Variable Computer, Electrical Engineering (Volume: 72, Issue: 2, February 1953) [Online] Available: <https://ieeexplore.ieee.org/document/6438502> [Accessed: June 14, 2019].

7.2 Pictures

[1] courtesy of U.S federal government <https://www.flickr.com/photos/departmentofenergy/11239892036/> [Accessed: June 14, 2019]

[2] J. Weidendorfer : IN0004. Vorlesung, Topic: "Einführung in die Rechnerarchitektur" Technische Universität München, München, Wintersemester 2017/2018. Folien Kapitel 1 (a) Seite 11

[3] https://commons.wikimedia.org/wiki/File:Harvard_architecture.svg [Accessed: June 14, 2019] (Adapted)

[4] https://www.theregister.co.uk/2015/09/17/boffins_reinvent_core_memory_ferroelectric/ [Accessed: June 14, 2019]

[5] Barry Fagin: Debugging on the Shoulders of Giants: Von Neumann's Programs 65 Years Later Computer (Vol-

ume: 45, Issue: 11, Nov. 2012) [Online]. Available: <https://ieeexplore.ieee.org/document/6152078> [Accessed: June. 14, 2019].

[6] https://en.wikipedia.org/wiki/Selectron_tube/media/File:Selectron_tube_p1270778.jpg [Accessed: June 14, 2019]

[7] https://en.wikipedia.org/wiki/Williams_tube/media/File:Williams_tube.agr.jpg [Accessed: June 14, 2019]

[8] Barry Fagin: Debugging on the Shoulders of Giants: Von Neumann's Programs 65 Years Later Computer (Volume: 45, Issue: 11, Nov. 2012) [Online]. Available: <https://ieeexplore.ieee.org/document/6152078> [Accessed: June. 14, 2019].

[9] [https://en.wikipedia.org/wiki/EDSAC/media/File:EDSAC_\(19\).jpg](https://en.wikipedia.org/wiki/EDSAC/media/File:EDSAC_(19).jpg) [Accessed: June 14, 2019]

[10] M.V. Wilkes, Olivetti and Oracle Res. Lab., Cambridge, UK: Arithmetic on the EDSAC IEEE Annals of the History of Computing (Volume: 19, Issue: 1, Jan-Mar 1997) [Online]. Available: <https://ieeexplore.ieee.org/document/560726> [Accessed: June 14, 2019].

[11] MARTIN CAMPBELL-KELLY: Programming the EDSAC: Early Programming Activity at the University of Cambridge IEEE Annals of the History of Computing (Volume: 20, Issue: 4, Oct-Dec 1998) [Online]. Available: <https://ieeexplore.ieee.org/document/728229> [Accessed: June. 14, 2019].

[12] Matsuura Tomoya, "Art Project" https://matsuuratomoya.com/en/works/post-past_sotsuten/ [Online]. Available: https://matsuuratomoya.com/en/works/post-past_sotsuten/ [Accessed: June. 14, 2019].

[13] <https://hackaday.com/2018/09/28/retrotechtacular-heres-how-they-programmed-the-edsac-computer/>

[14] <http://www.chilton-computing.org.uk/acl/literature/chapman/p015.htm> [Accessed: June 14, 2019]

[15] M.V. Wilkes: Early computer developments at Cambridge: The EDSAC Radio and Electronic Engineer (Volume: 45, Issue: 7, July 1975) [Online]. Available: <https://ieeexplore.ieee.org/document/5268249> [Accessed: June 14, 2019].

[16] courtesy of the U.S. Army <https://commons.wikimedia.org/wiki/File:Edvac.jpg> [Accessed: June 14, 2019]

[19] M.R. Williams: The origins, uses, and fate of the ED-VAC, IEEE Annals of the History of Computing (Volume: 15, Issue: 1, 1993) [Online]. Available: <https://ieeexplore.ieee.org/document/194089> [Accessed: June 14, 2019] Page 28, translated

[20] Gluck - Electronic Discrete Variable Computer, February 1953 (taken from Paper [12])

[21] <https://gizmodo.com/delay-line-memory-how-computers-remembered-before-ram-5498229> [Accessed: June 14, 2019]

[23] MARTIN CAMPBELL-KELLY: Programming the ED-SAC: Early Programming Activity at the University of Cambridge IEEE Annals of the History of Computing (Volume: 20, Issue: 4, Oct-Dec 1998) [Online]. Available: <https://ieeexplore.ieee.org/document/728229> [Accessed: June. 14, 2019].