DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

T3_3100 Studienarbeit I

Duale Hochschule Baden-Württemberg (DHBW)
Mannheim

Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers zu Lehrzwecken mittels einer Software zur Simulation von digitalen Schaltungen

von Silas Gerhard

Fakultät Technik

Studiengang Elektrotechnik (B.Eng.)

Fachrichtung Elektrische Energietechnik

Studienkurs: TEL18-AET

Studiengangsleitung: Prof. Dr.-Ing. Nicole Möhring, DHBW

Bearbeitungszeitraum: 18.01.2021 – 05.04.2021

Studierender: Silas Gerhard

Matrikelnummer: 4010210

Betreuer: Dipl.-Ing. Gerhard Lehmann, DHBW

Diese Studienarbeit ist verfasst, innerhalb des Studiums zum Bachelor of Engineering (B.Eng.) im Studiengang Elektrotechnik (Elektrische Energietechnik) an der Dualen Hochschule Baden-Württemberg (DHBW), von Silas Merlin Gerhard geboren am 22.10.2000 in Aalen (D).

Dualer Student und Verfasser der Arbeit
Silas Gerhard

Wiesbaden, 05.04.2021

Ort und Datum

Erklärung zur eigenständigen Verfassung

Bezugnehmend auf die "Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg (DHBW) (Studien- und Prüfungsordnung DHBW Technik – StuPrO DHBW Technik) 29. September 2017" 2. Kapitel §5 (3),

versichere ich, dass diese Studienarbeit (T3_3100) mit dem Titel „Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers zu Lehrzwecken mittels einer Software zur Simulation von digitalen Schaltungen" nur von mir selbst und nur unter Zuhilfenahme der angegebenen Quellen und Hilfsmitteln verfasst wurde. Die digitale Version und die gedruckte Version dieses Dokuments sind inhaltlich gleich.

Dualer Student und Verfasser der Arbeit
Silas Gerhard

Wiesbaden, 05.04.2021

Ort und Datum

Vorwort

Diese Arbeit ist eine Studienarbeit im 5. Semester meines Studiums zum Bachelor of Engineering (B.Eng.) im Studiengang Elektrotechnik mit Studienrichtung Elektrische Energietechnik.

Auch wenn das Thema dieser Studienarbeit nicht aus der Energietechnik kommt, hat es mir doch sehr viel Freude bereitet, daran zu arbeiten. Vor allem mein Verständnis für Mikroprozessoren und Microcomputern wurde dadurch erweitert und gefestigt. Einerseits erkennt man, dass ein Mikroprozessor schon mit einer recht überschaubaren Anzahl von logischen Schaltungen realisierbar ist. Auf der anderen Seite jedoch wird auch klar, welche hohe Komplexität in einzelnen Komponenten vorhanden sein muss, damit das Gesamtkonstrukt funktionieren kann. Vor allem die gleichzeitige Koordination von Prozessablauf, Schaltungstakt und Gatterlaufzeiten ist hierbei herausfordernd. Nicht zuletzt die intensive Beschäftigung mit der Interruptverarbeitung als auch das Auseinandersetzen mit einem Simulationsprogramm für digitale Schaltungen haben mir einen Einblick in die Hintergründe der digitalen Schaltungsentwicklung gegeben.

In jedem Fall hat diese Studienarbeit die Prozesse innerhalb eines Mikrocomputers greifbarer gemacht und mein Interesse für diesen Bereich der Elektrotechnik nachhaltig geweckt.

Ich möchte mich an dieser Stelle auch bei meinem Betreuer, Herrn Dipl.-Ing. Gerhard Lehmann, für die effektive und angenehme Betreuung bedanken.

Kurzfassung

Diese Studienarbeit beschäftigt sich mit der Interruptverarbeitung eines Mikrocomputers. Im konkreten soll die Hardware-Interruptverarbeitung eines Mikrocomputers mit einem 8086 Mikroprozessor des Herstellers Intel mittels Software visualisiert werden. Da diese Visualisierung zu Lehrzwecken genutzt werden soll, ist nur der prinzipielle Ablauf aufgezeigt und keine prozessortaktgetreue Nachbildung entwickelt. Dabei werden auch pädagogische Aspekte sowie vor allem die optische Aufbereitung der Visualisierung betrachtet. Umgesetzt wird die Visualisierung der Hardware-Interruptverarbeitung als Simulation mit einer Software zur Simulation von digitalen Schaltkreisen. Durch die Entwicklung von vereinfachten Bausteinen werden die für die Hardware-Interruptverarbeitung notwendigen Funktionen von Prozessor, RAM und Interruptcontroller nachgebildet. Diese Bausteine werden dann in der obersten Ebene verschaltet und optisch aufbereitet. Um das Verständnis des Ablaufs zu vereinfachen und zeitunabhängig zu gestalten, erfolgt die Verarbeitung im Einzelschrittmodus. Dieses Verfahren ermöglicht es dem Benutzer der Visualisierung, die einzelnen Schritte der Hardware-Interruptverarbeitung Stück für Stück nachzuvollziehen. Das Studienprojekt verläuft erfolgreich und kann am Ende ein Produkt aufweisen, dass alle Kernziele erfüllt. Die simulative Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers kann gerade in der digitalen Lehre vorteilhaft sein.

Abstract

This student research project deals with the interrupt processing of a microcomputer. Specifically, the hardware interrupt processing of a microcomputer with an 8086 microprocessor from the manufacturer Intel is to be visualized by a software. Since this visualization is to be used for teaching purposes, only the principal sequence is shown, and no processor clock-faithful reproduction is developed. Pedagogical aspects as well as especially the optical preparation of the visualization are also considered. The visualization of the hardware interrupt processing is implemented as a simulation with a software for the simulation of digital circuits. By developing simplified building blocks, the functions of processor, RAM, and interrupt controller necessary for hardware interrupt processing are simulated. These building blocks are then interconnected in the top level and visually prepared. To simplify the understanding of the process and to make it time independent, the processing is done in single step mode. This procedure enables the user of the visualization to follow the individual steps of the hardware interrupt processing step by step. The project runs successfully and can show a product at the end that fulfills all core objectives. The simulative visualization of the hardware interrupt processing of a microcomputer could achieve a benefit especially in digital teaching.

Inhaltsverzeichnis

1 Anhangsverzeichnis

2 Abbildungsverzeichnis

3 Abkürzungen und Definitionen

(In alphabetischer Reihenfolge)

ALU

Arithmetic **L**ogic **U**nit, Arithmetisch Logische Einheit eines Mikroprozessors.

BIU

Bus **I**nterface **U**nit, Bus-Schnittstellen-Einheit. Prozessorteileinheit eines Intel 8086 Mikroprozessors, welche die Kommunikation zum externen Systembus unterhält.

CPU

Central Processing Unit, Zentrale Verarbeitungseinheit. Entspricht einem Mikroprozessor.

CS

Code**s**egment.

DIL

Dual-**I**n-**L**ine, zweireihig. Ein DIL-Gehäuse ist eine längliche Gehäuseform für integrierte Schaltungen die zwei Reihen von Anschlussstiften besitzt.

ETVX

Entry **T**ask **V**erification & **V**alidation E**x**it ist ein Lebenszyklusmodell, das auch Elementarprozessschema genannt wird.

EU

Execution **U**nit, Ausführungs-Einheit. Ausführende Prozessorteileinheit eines Intel 8086 Mikroprozessors.

GPL

General **P**ublic **L**icense. Lizenz für Softwareprodukte die kostenfrei erhältlich ist, jedoch z.B. keine Garantieabdeckung enthält.

IF

Interrupt-**E**nable-**F**lag, Interrupt-Freigabe-Markierung.

INTA

Interruptacknowledge. Ausgang am Intel 8086 Mikroprozessor zur Interruptannahme bzw. Signalleitung eines Mikrocomputers mit 8086 Prozessor.

INTR

Interrupt. Interrupteingang am Intel 8086 Mikroprozessor bzw. Signalleitung eines Mikrocomputers mit 8086 Prozessor.

IP

Instruction **P**ointer, Befehlszeiger.

IRET

Interrupt **Ret**urn, Interrupt-Rückkehr. Befehl am Ende einer ISR zur Rückkehr zum Hauptprogramm.

ISR

Interrupt-**S**ervice-**R**outine. Routine zur Behandlung eines Interrupts inklusive Unterprogramm.

KPI

Key **P**erformance **I**ndicator, Schlüsselleistungsindikator.

MiB

Megab**i**nary**b**yte. Der Präfix Megabinary entspricht einem Wert von 2^{20} im Gegensatz zum Präfix Mega, der einem Wert von 10^6 entspricht. Entsprechend gleicht ein Megabinarybyte 2^{20} Byte.

Nesting

Nesting beschreibt das Ineinandereinbetten von Interrupts, also das Annehmen von weiteren Interrupts innerhalb einer ISR.

NMI

Non-Maskable-Interrupt, Nicht-Maskierbarer-Interrupt.

RAM

Random **A**ccess Memory, Arbeitsspeicher.

SP

Stack **P**ointer, Stapelzeiger.

SS

Stack**s**egment.

4 Einführung

4.1 Aufgabenstellung

Die Hardware-Interruptverarbeitung eines Mikrocomputers ist für Studierende nicht immer direkt und vollständig verständlich. Dies lässt sich vor allem durch die vielen aufeinanderfolgenden Schritte und die Tatsache, dass mehrere Komponenten (Prozessor, RAM, Interruptcontroller) daran beteiligt sind begründen. Daher ist es sinnvoll den Prozess der Hardware-Interruptverarbeitung zu visualisieren um das Verständnis durch effektive optische Unterstützung zu fördern.

Da die Lehreinheiten Beispielhaft anhand des Intel 8086 Prozessors durchgeführt werden, soll auch die Visualisierung der Hardware-Interruptverarbeitung sich an der Interruptverarbeitung des 8086 Prozessors orientieren.

Die Aufgabenstellung und somit auch das Ziel der Studienarbeit ist die Entwicklung einer Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Prozessors zu Lehrzwecken. Die Visualisierung soll nach Möglichkeit in Einzelschritttaktung durchlaufen werden können um Schritt für Schritt den Ablauf zu erklären. Die Umsetzung soll mittels einer Software erfolgen.

Weitere Details zu den Anforderungen und das Projektkonzept werden im nächsten Kapitel erläutert.

4.2 Auswahl der Software

Die Art der verwendeten Software sowie die Software im speziellen hat großen Einfluss auf das Ergebnis, da diese die Realisierungsgrenzen für die Visualisierung setzt und außerdem den Grundstein für die Implementierung von Funktionen legt. Daher ist es wichtig, bei der Auswahl der Software die Vor- und Nachteile gründlich abzuwägen sowie die möglichen Risiken zu betrachten.

Zur Auswahl stehen bei dieser Aufgabenstellung grundsätzlich zwei verschiedene Typen von Software. Zum einen eine Software, die sich auf das Optische konzentriert und somit nur eine vorgegebene Abfolge visuell wiedergibt. Dieses Konzept wird zum Beispiel von einer

Animationssoftware angewendet. Der andere Typ ist eine Software die simulativ arbeitet und eine Modellierung von technischen Komponenten möglich macht.

Der Vorteil einer Animationssoftware ist vor allem, dass hier das Optische im Fokus steht. Es gibt viele Möglichkeiten die Interruptverarbeitung mit visuellen Stilmitteln auszugestalten, um eine maximale Förderung des Verständnisses zu erreichen. Außerdem vereinfacht diese Variante die Umsetzung enorm, da nur ein bestimmter Ablauf programmiert werden muss, der dann abgespielt werden kann. Der Einzelschritt wäre in diesem Fall durch Pausieren der Animation gegeben. Nachteilig ist jedoch, dass bei einer solchen Simulation keine allgemeine Anwendbarkeit ohne großen Aufwand implementiert werden kann. So kann man immer wieder nur dieselbe Animation abspielen und hat wenig Möglichkeit, eine aktive Benutzerinteraktion zu ermöglichen. Dies wäre jedoch pädagogisch von nutzen.

Die Simulation ist in der Lage, genau diese Defizite der Animation bereitzustellen. Modelliert man in einer Simulationssoftware eine bestimmte Struktur, so kann man dabei Parameter definieren, die vom Benutzer selbst gewählt und bei jedem Durchlauf neu definiert werden können. Die Visulisierung gewinnt dadurch an Allgemeingültigkeit. Beispielsweise kann man so eine Hardware-Interruptverarbeitung für mehrere Interrupts ausgehend von verschiedenen Registerinhalten realisieren. Bei einer Animation wären hier nur einige Beispiele möglich, da ansonsten der Aufwand zu groß werden würde. Außerdem wäre die Visualisierung jeweils statisch auf eine bestimmte Einstellung pro Durchlauf festgesetzt. Der Nachteil einer Simulation liegt darin, dass eine entsprechende Software ggf. nur mangelhafte Visualisierungsmöglichkeiten anbietet. Die optische Aufbereitung gestaltet sich somit im Allgemeinen wesentlich komplizierter als bei einer Animation. Des Weiteren muss bei dieser Variante unter Umständen eine viel konkretere Modellierung der technischen Komponenten erfolgen. Der Aufwand orientiert sich dabei daran, welche Hilfsmittel der Simulator zur Verfügung stellt.

Als Animationssoftware hat sich vor allem Adobe Animate angeboten. Als hier jedoch technische Probleme bei der Anwendung der Software aufgetreten sind, ist aus diesem Grund und infolge der oben genannten Nachteile von Animationssoftware die Entscheidung auf den Einsatz eines Simulators gefallen. Außerdem ist der Erwerb von Adobe Animate mit Kosten verbunden, wohingegen die letztendlich ausgewählte Software kostenfrei erhältlich ist.

Um eine gewisse Interaktion des Benutzers mit der Visualisierung zu ermöglichen und die technischen Abläufe hinter der Hardware-Interruptverarbeitung besser darstellen zu können, wird schließlich eine Simulationssoftware eingesetzt. Dabei stehen einige Programme zur Auswahl, welche diese Ziele erfüllen können. Einer dieser Simulatoren zeichnet sich dadurch aus, dass er nur für die Simulation von digitalen Schaltkreisen vorgesehen ist, gute optische Darstellungsmöglichkeiten bietet und sehr einfach zu bedienen ist. Außerdem sind bereits Erfahrungen im Umgang mit diesem Simulator vorhanden.

Der Name der verwendeten Software lautet „Digital“. Sie ist von Helmut Neemann, einem Professor an der Dualen Hochschule Baden-Württemberg (DHBW) in Mosbach entwickelt worden. Digital ist ein Simulator rein für digitale Schaltkreise und bietet eine Vielzahl an Möglichkeiten, welche bei der Umsetzung der Visualisierung teilweise genutzt werden und die Aufgabe erleichtern. [1]

Der Simulator selbst und seine Möglichkeiten werden in späteren Kapiteln noch thematisiert.

5 Projektkonzept

Die Studienarbeit wird wie ein Projekt behandelt und anhand eines Projektkonzeptes bearbeitet. Dieses Projektkonzept soll im Folgenden vorgestellt werden. Es ist als eigenständiges Dokument ebenfalls in *Anhang 1 – Projektkonzept* wiederzufinden.

5.1 Anforderungsspezifikation

Die Anforderungsspezifikation definierte die Ziele eines Projekts sowie in welchem Umfang diese zu erreichen sind. Die Anforderungsspezifikation befindet sich als eigenständiges Dokument im *Anhang 2 – Anforderungsspezifikation* und wird außerdem im Folgenden behandelt.

5.1.1 Definition der Kernziele (Festforderungen)

Für das Projekt der Visualisierung der Hardware-Interruptverarbeitung sind folgende Festforderungen definiert.

- Entwicklung einer Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers

- Anpassung der Entwicklung auf Lehrzwecke (pädagogische Verwendbarkeit, verständliche und nachvollziehbare Erklärungen)
- Die Visualisierung soll digital medial mithilfe einer Software durchgeführt werden
- Die Interruptverarbeitung soll nach außen hin der des Intel 8086 Prozessors entsprechen

5.1.2 Optionale Ziele (Wünsche)

Optionale Ziele zeichnen sich dadurch aus, dass sie nicht erreicht werden müssen, sondern abhängig von den gegebenen Möglichkeiten und Umständen umgesetzt werden. Auch wirtschaftliche Aspekte (Zeit, Kosten, etc.) spielen hier eine Rolle. Folgende optionale Ziele sind festgelegt.

- Die Visualisierung sollte nach Möglichkeit in Einzelschritttaktung durchlaufen werden können
- Falls durchführbar auch Darstellung der Prozesse innerhalb des Prozessors
- Vorzugsweise Verwendung des Intel 8086 Prozessors
- Implementierung von Übungsaufgaben
- So weit wie möglich als gekapselte Anwendung ausführbar

5.1.3 Dateien und Dokumente

Des Weiteren wird definiert, welche Ergebnisse des Projekts konkret in welchem Format vorliegen müssen. Dazu wird definiert, welche Dateien und Dokumente und in welcher Form diese erstellt werden müssen.

Folgende Dateien bzw. Dokumente sind zum Abschluss des Projektes in folgendem Umfang notwendig.

Tabelle 1 – Spezifikation der Dateien und Dokumente für den Projektabschluss

Name	Beschreibung	Dateiformat
Visualisierung der Hardware-Interruptverarbeitung	Fertige Visualisierung der Hardware-Interruptverarbeitung eines Intel 8086 Prozessors nach den Spezifikationskriterien	Anwendung bzw. Simulation innerhalb einer Anwendung
Anleitung zur Verwendung der Visualisierung	Beschreibung der Funktionsweise und Handhabung der entwickelten Anwendung/Simulation	PDF

5.1.4 Risiken

Da es unwahrscheinlich ist eine Software zu finden, welche alle Anforderungen exakt abbildet und der Zeitaufwand zur Programmierung einer komplett neuen Anwendung zu groß wäre, ist zu erwarten, dass vereinzelt Forderungen nicht komplett erfüllt werden können bzw. teilweise etwas weniger elegante Lösungen herangezogen werden müssen.

5.1.5 Testung

Die Testung der Visualisierung soll in einzelnen Komponenten sowie als zusammengefügtes Gesamtkonzept erfolgen. Dabei wird hinsichtlich verschiedener Kriterien wie Funktionalität, Realitätsbezug, Verständlichkeit und Handhabung getestet.

5.2 Projektplanung

Die Projektplanung untergliedert sich in zwei Teile. Zum einen in die Ablaufplanung anhand eines Lebenszyklusmodells und zum anderen in die zeitliche Planung. Beide werden im Folgenden erläutert.

5.2.1 Ablaufplanung und Lebenszyklusmodell

Grundsätzlich wird eine agile Projektentwicklung angestrebt. Die Kernzeile des Projekts bleiben im Fokus der Entwicklung und werden entsprechend der Rahmenbedingungen angepasst. Auf Herausforderungen wird situationsgerecht und interaktiv reagiert. Angestrebt sind schnelle und effiziente Lösungen im Sinne des Gesamtkonzepts.

Das Projekt soll anhand des Elementarprozessschemas ETVX bearbeitet werden. Dieses Lebenszyklusmodell wird anhand von Phasen inkrementell abgearbeitet. *Abbildung 1* zeigt den prinzipiellen Ablauf innerhalb des Elementarprozessschemas.

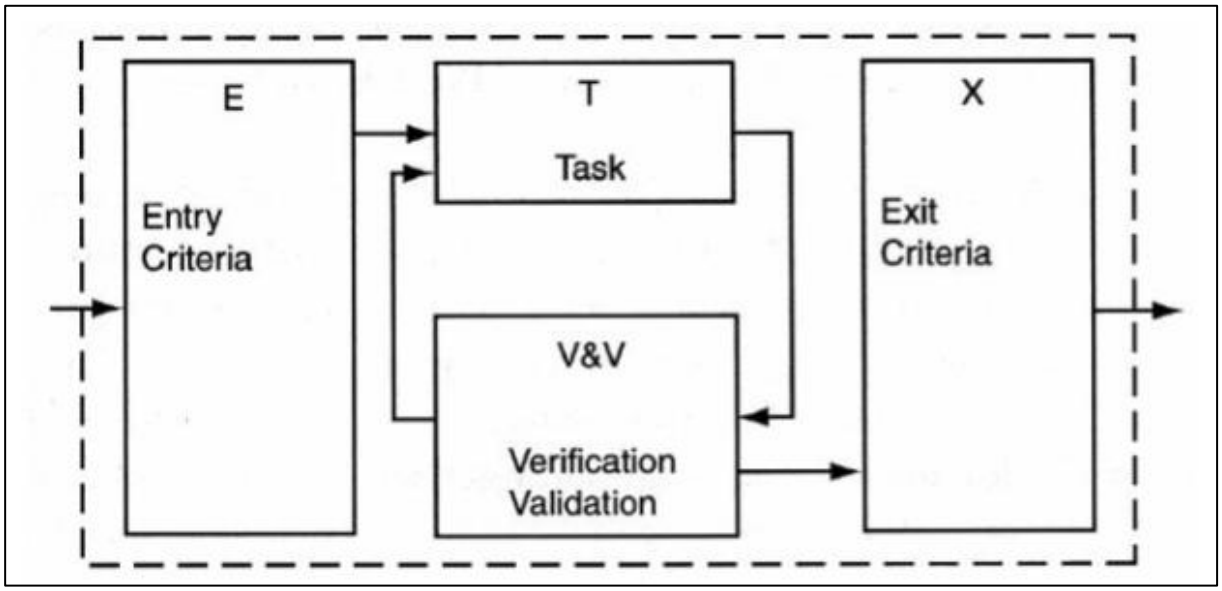


Abbildung 1 – Schema zum prinzipiellen Ablauf des Elementarprozessschemas ETVX [2, S. 24]

Die Bezeichnung ETVX kommt aus dem Englischen und steht für Entry, Task, Verification & Validation und Exit.

Jede Projektphase wir dabei nach dem in *Abbildung 1* dargestellten Schema abgearbeitet. Zunächst werden alle Eingangskriterien (Entry Criteria) geprüft, die notwendig sind, um in die jeweilige Phase einzutreten. Daraufhin wird die eigentliche Aufgabe (Task) bearbeitet. Durch die Validierungsphase wird die Qualitätssicherung direkt mit der Aufgabe verknüpft. Diese Teilphase soll feststellen, ob alle Entwicklungsschritte exakt ausgeführt, die Eingangsdokumente richtig und vollständig weiterbearbeitet wurden und die Kundenanforderungen erfüllt werden. Erst bei Erfüllung aller Ausgangskriterien (Exit Criteria) wird der jeweilige Elementarprozess beendet und der nächste kann in Angriff genommen werden. Wesentliche Idee hinter diesem Modell ist es, jeden Arbeitsschritt unmittelbar mit einer Qualitätssicherungsteilphase zu verbinden. Fehler können so früh gefunden und relativ kostengünstig behoben werden. Es besteht jedoch die Gefahr eines Stille-Post-Effekts bei mehrstufiger Weitergabe von Informationen. *Tabelle 1*, *Tabelle 2* und *Tabelle 3* definieren die Prozessphasen für dieses Projekt nach dem ETVX Schema. [2]

Tabelle 2 – Projektphasen I und II nach dem Elementarprozessschema ETVX

Phase	I	II
Name	Projektvorbereitung	Entwurf
Eingangskriterium	Mit Beginn des Projekts	Abschluss von Phase I
Aufgabe	Anforderungsspezifikation und Zeitplan erstellen, Auswahl der Software, Festlegung des Projekttitels, Vorbereitung der Dokumente, Literaturrecherche	Ablauf der Interruptverarbeitung im Diagramm darstellen, Austesten der Fähigkeiten der Software inkl. Feststellung von Problembereichen, Ablauf der Entwicklung festlegen
Verifikation	Sind Anforderungen und Zeitplan realistisch umsetzbar? Ist die Software zum Erreichen der Kernziele sinnvoll? Spiegelt der Projekttitel die Ziele wieder? Mindestens 3 passende Quellen gefunden?	Entspricht die Darstellung der tatsächlichen Interruptverarbeitung und enthält sie alle wichtigen Informationen? Decken die erkannten Funktionen der Software alle Anforderungen der Interruptverarbeitung ab? Ist der Arbeitsaufwand realistisch und absehbar?
Ausgangskriterium	Die Aufgaben sind hinsichtlich der Validierungskriterien erfüllt. Es sind keine Frage mehr offen bezüglich der Aufgabenstellung und der Ziele.	Der Ablauf der Interruptverarbeitung in der Software ist vollständig festgelegt. Es besteht eine genaue Vorstellung über die Entwicklung der Simulation.

Tabelle 3 – Projektphasen III und IV nach dem Elementarprozessschema ETVX

Phase	III	IV
Name	Entwicklung I	Entwicklung II
Eingangskriterium	Abschluss von Phase II	Abschluss von Phase III
Aufgabe	Entwicklung der einzelnen Komponenten die für die Interruptverarbeitung notwendig sind (Interruptcontroller, RAM, ROM, Verbindungen und Peripherie, CPU), Zwischentestung der einzelnen Komponenten, Planung des Zusammenbaus	Zusammenbau der Komponenten und Entwicklung der Schnittstellen, Optimierungen, um Einzelschritttaktung zu ermöglichen, Hinzufügen von erklärenden Elementen, Optionale Features
Verifikation	Funktionieren die Komponenten, wie sie sollten? Ist die Umsetzung anschaulich und optisch ansprechend (ggf. eigene Graphik erstellen)? Sind Komplexität und schnelle Verständlichkeit ausgewogen? Gibt es Probleme beim Zusammenbau?	Funktioniert alles nach der Zusammensetzung noch wie zuvor? Fehlen Bauteile oder erfüllen Schnittstellen nicht ihren Zweck? Ist der Ablauf verständlich und erkennbar?
Ausgangskriterium	Alle Komponenten sind entwickelt und funktionstüchtig. Dem Zusammenbau steht nichts mehr im Wege.	Die Simulation bildet die Hardware-Interruptverarbeitung funktionsgetreu ab und ist verständlich. Die optische Entwicklung ist ebenfalls abgeschlossen. Mögliche Zusatzfunktionen sind ausgreift.

Tabelle 4 – Projektphasen V und VI nach dem Elementarprozessschema ETVX

Phase	V	VI
Name	Testung	Projektabschluss
Eingangskriterium	Abschluss von Phase IV	Abschluss von Phase V
Aufgabe	Überprüfung der Simulation durch Tests hinsichtlich Funktionalität, Verständnis, Handhabung und Fehleranfälligkeit, Testung mit und ohne Einzelschrittmodus	Ausarbeitung von Anleitung und ggf. Übungsaufgaben, Organisatorische Aufarbeitung aller Projektinhalte, Überprüfung, ob die Projektziele unter Einhaltung der Vorgaben erreicht wurden, Feststellung von Entwicklungspotential und Nachlässigkeiten
Verifikation	Welche Fehler treten auf und wie können Sie behoben oder umgangen werden? Wird die Interruptverarbeitung den Studierenden durch die Visualisierung schneller verständlich?	Ist die Anleitung benutzerfreundlich? Ist alles erledigt? Was fehlt bzw. muss noch gemacht werden? Welche Verbessrungen wären angebracht? Was ist die größte Schwachstelle des Projekts?
Ausgangskriterium	Schwachstellen der Simulation sind erkannt und ggf. behoben. Im Regelfall funktioniert das Produkt wie es soll.	Das Projekt ist vollständig abgewickelt und es stehen keine Punkte mehr aus, die zur Erfüllung der Aufgabenstellungen notwendig sind. Das Projekt wurde kritische reflektiert und Verbessrungsmöglichkeiten sind bekannt.

5.2.2 Zeitplanung

Das Projekt soll so schnell wie möglich, jedoch mit korrekter Ausführung aller Prozessschritte abgeschlossen werden. Dabei soll möglichst der erstelle Zeitplan eingehalten werden. Der Zeitplan ist nach den einzelnen Phasen des Elementarprozessschemas strukturiert und weißt diesen Bearbeitungszeiträume zu. Der Zeitplan ist im Zeitleistenformat in *Anhang 3 – Zeitplan* verfügbar.

5.2.3 Schlüsselleistungsindikatoren – Key Performance Indicators (KPIs)

Die sogenannten Schlüsselleistungsindikatoren oder auch Leistungskennziffern sind meist unter ihrem englischen Namen „Key Performance Indicators“, abgekürzt KPIs, bekannt. Kennwerte, Eigenschaften und Zwischenergebnisse werden als Schlüsselleistungsindikatoren definiert, wenn sie eine große Aussagekraft über den Erreichungsgrad des Gesamtprozesses oder des Gesamtprojekts haben. Folgende Schlüsselleistungsindikatoren sind für dieses Projekt festgelegt.

- Der Zeitplan wird nicht gedehnt (Die Bearbeitungszeit einer Phase ist kleiner als eine Woche)
- Die Schnittstelle zwischen Bit- und Hexadezimalsystem funktioniert
- Eingabe und Ausgabe funktionieren ordnungsgemäß
- Das Einbetten der Schaltungen verläuft erfolgreich
- Die Simulation ist ohne Fachwissen bedienbar
- Ein Interrupt lässt sich in der Visualisierung ohne Probleme begutachten und wird von Externen verstanden

6 Stand von Wissenschaft und Technik

6.1 Der Intel 8086 Mikroprozessor

Der 8086 ist der erste 16-Bit Prozessor von Intel. Seine ALU und Register sind somit 16-Bit groß. Mit dem 20-Bit Adressbus des 8086 ist es möglich, 1 MiB direkt zu adressieren. Durch verschiedene Varianten des 8086 sind Taktfrequenzen von mindestens 2 MHz bis zu 10 MHz möglich. Diese Prozessorkennwerte liegen natürlich weit hinter der heutigen Prozessortechnik

zurück und entsprechen somit nicht mehr dem Stand der Technik. Das liegt daran, dass der 8086 schon 1978 von Intel entwickelt wurde. Durch seinen genügend einfachen Aufbau und seine Betriebsweise eignet sich der 8086 jedoch gegenüber heutigen Hochleistungsprozessoren sehr gut für Lehrzwecke und kommt daher an Hochschulen noch häufig zum Einsatz. In diesem Sinne entspricht der 8086 also immer noch einem gewissen Stand der Technik. Außerdem ist das Verständnis des 8086 Grundlage für das Verständnis der gesamten Intel x86 Prozessorreihe. *Abbildung 2* zeigt einen Intel 8086 Mikroprozessor im DIL-Gehäuse. [3, S. 367] [4, S. 63] [5, Kap. 1]

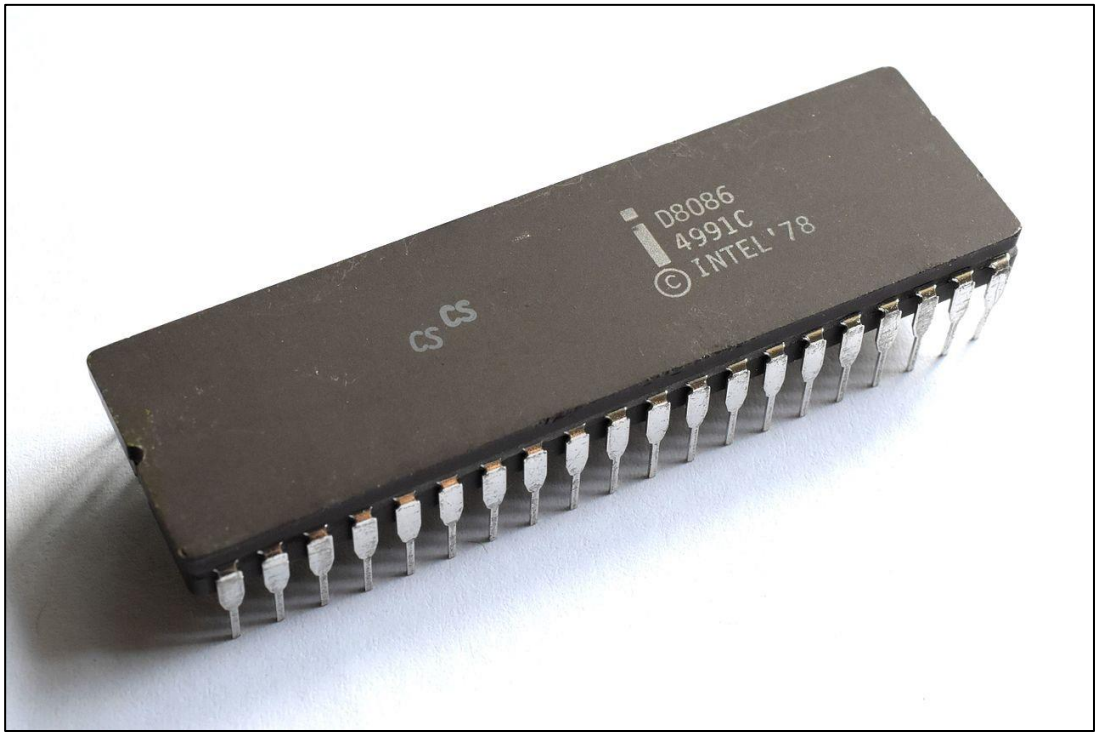


Abbildung 2 – Intel 8086 Mikroprozessor im DIL-Gehäuse [6]

Abbildung 3 zeigt einen Blockschaltplan und die Anschlüsse des 8086 Prozessors. Grundsätzlich lässt sich dieser in zwei Bereiche unterteilen. Die Bus-Schnittstellen-Einheit (BIU – Bus Interface Unit) koordiniert die Kommunikation mit dem externen Systembus und enthält die Segmentregister und die Befehlswarteschlange. Die Ausführungs-Einheit (EU – Execution Unit) enthält die Daten- und Adressregister sowie die ALU und ist das Kernstück des Prozessors. Diese Aufteilung des Prozessors erlaubt höhere Prozessgeschwindigkeiten im Vergleich zu Vorgängermodellen. Der Adress- und Datenbus sowie ein Teil der Steuerleitungen werden im Zeitmultiplex betrieben. Da die Adressregister nur 16-Bit breit sind, wird mithilfe

der Segmentregister und einer Offsetaddition eine 20-Bit breite Speicheradresse gebildet. Dabei wird das Segmentregister mit 10h multipliziert (also um eine Stelle nach links verschoben - Offset) und dann das Adressregister bzw. der Zeiger dazu addiert. Durch diese Operation ergibt sich ein 20-Bit Wert. [4, S. 63]

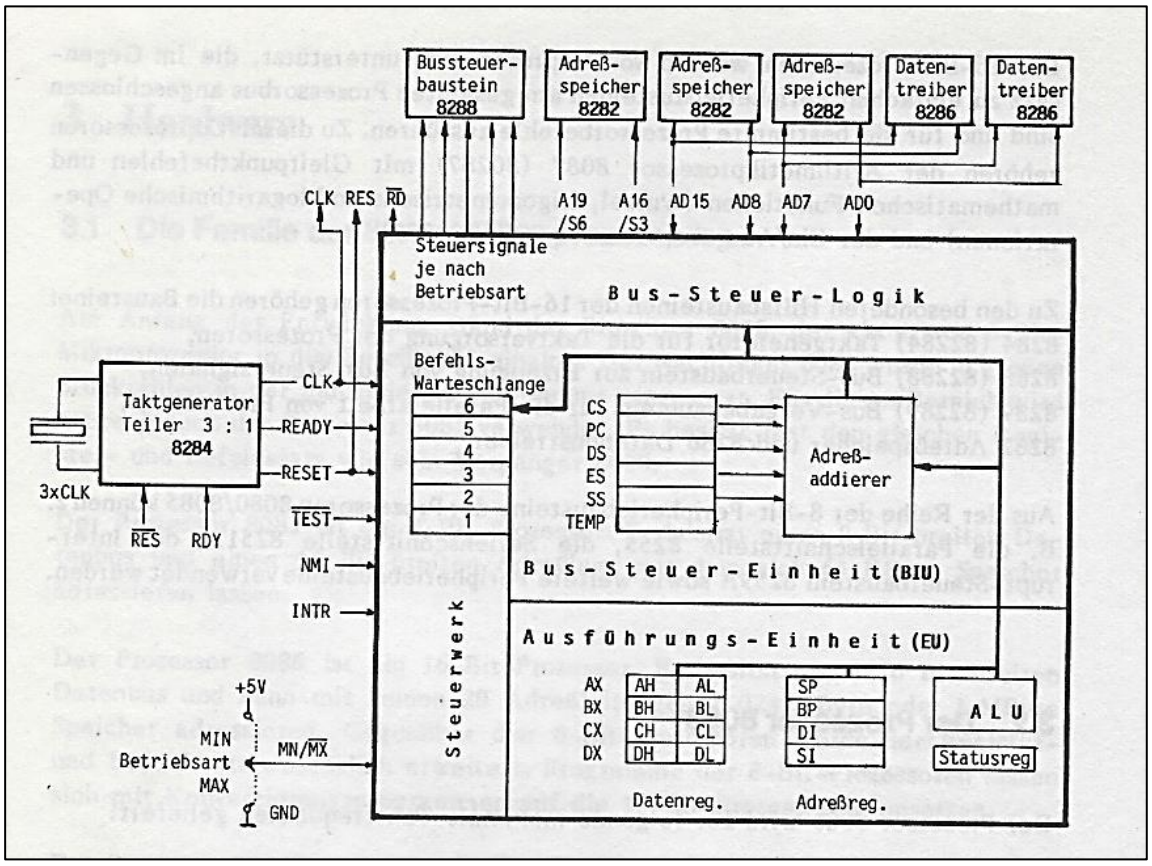


Abbildung 3 – Blockschaltbild und Anschlussübersicht eines Intel 8086 Mikroprozessors [4, S. 64]

6.2 Interruptverarbeitung von Mikroprozessoren

Ein Interrupt bietet die Möglichkeit, ein Programm von außen zu unterbrechen. Basierend auf diesem Prinzip werden Tastatureingaben an einem Computer bearbeitet oder besondere Ereignisse bei Prozessoren in der industriellen Anwendung behandelt. Wird ein Interrupt ausgelöst, so unterbricht der Prozessor das aktuell laufende Programm und führt eine für den Interrupt spezifische Interrupt-Service-Routine (ISR) aus. Anschließend wird die Bearbeitung des unterbrochenen Programms fortgesetzt. Um dies zu ermöglichen, muss vor der Abarbeitung der ISR die Rücksprungadresse des laufenden Programms auf den Stack gerettet werden. Daher ist es katastrophal, sollte ein Interrupt eintreten, bevor der Stack initialisiert wurde. Um Problemen dieser Art vorzubeugen, ist es möglich, die Annahme von Interrupts zu

sperren. Diese werden dann erst freigegeben, sobald der Stackpointer (SP) initialisiert ist. [7, S. 266]

6.3 Die Hardware-Interruptverarbeitung des 8086 Prozessors

Es gibt grundsätzlich zwei Typen von Interrupts. Der Software-Interrupt wird von einem Programm oder dem Prozessor selbst ausgelöst. Dieser wird hier nicht weiter behandelt. Der Hardware Interrupt wird durch ein von außen kommendes Signal ausgelöst. Seine Verarbeitung bei 8086 soll im Folgenden behandelt werden.

Wie *Abbildung 3* zu entnehmen ist, verfügt der 8086 über zwei Eingänge durch die externe Hardware-Interrupts angenommen werden können. Diese tragen die Namen NMI (Non-Maskable-Interrupt) und INTR (Interrupt). Durch Hilfsmarkierungen wie beispielsweise die Interrupt-Freigabe-Markierung (Intrrupt-Enable-Flag - IF) ist es programmseitig möglich, Interrupts zu maskieren und somit deren Ausführung bei Eintreten zu verhindern. Lediglich bei Interrupts, die über den INTR Eingang angeboten werden, ist eine Maskierung möglich. Beim NMI Eingang ist dies, wie der Name schon sagt, nicht möglich. An diesen können z.B. externe Überwachungsgeräte (Watchdogs) angeschlossen werden. Der NMI wird im Folgenden nicht weiter betrachtet. [8, 2-22 - 2-29]

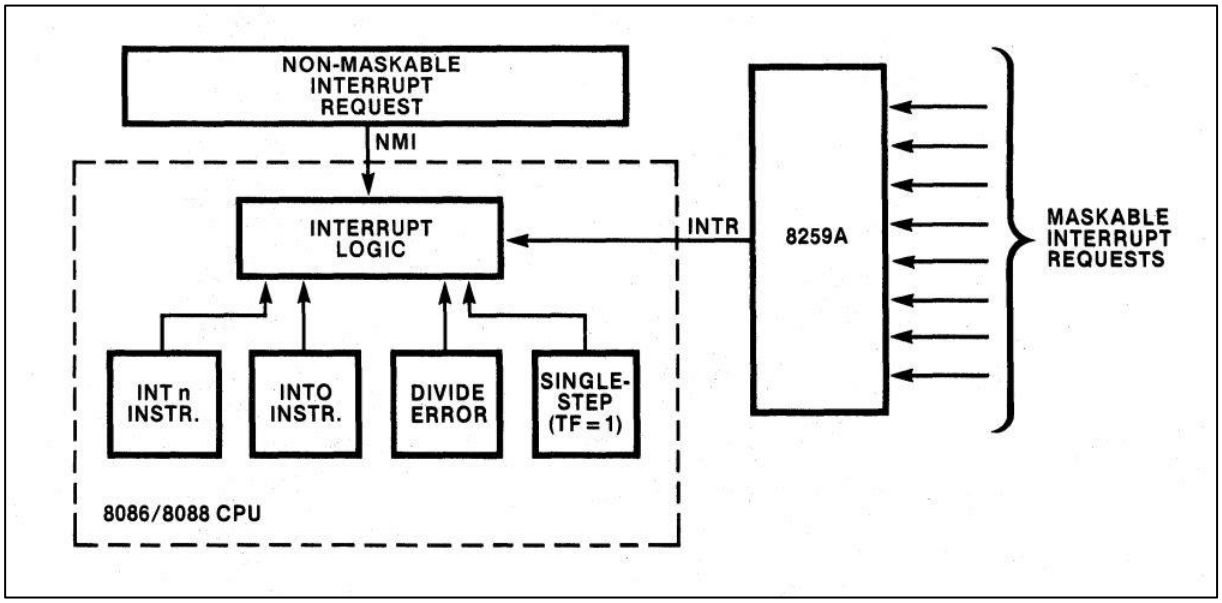


Abbildung 4 – Prinzipschaubild zu den Interrupt-Quellen eines Intel 8086 Mikroprozessors inkl. Interruptcontroller [8, S. 2-22]

Um die Anzahl der Anschlusssteifte am Prozessor nicht unnötig zu erhöhen und diesen nicht mit der Interruptvorverarbeitung zu belasten, ist diese beim 8086 in einen Interruptcontroller ausgelagert. Die Interruptanfrage an den Prozessor findet dann durch den Interruptcontroller über die INTR-Leitung statt. Der INTR-Eingang des 8086 Prozessors ist nicht speichernd aber zustandsgesteuert, daher muss das INTR-Signal so lange gehalten werden, bis der Interrupt entweder angenommen oder quittiert wird. [8, 2-22 - 2-29]

Der 8086 Prozessor ist in der Lage, mit 256 verschiedenen Interrupts umzugehen. Wird ein Interrupt angenommen, so legt der Prozessor zuerst die Inhalte des CS- und IP-Registers auf den Stack, um nach der Interruptverarbeitung an der Stelle fortzufahren, an der er unterbrochen wurde. Durch den INTA-Ausgang (Interrupt Acknowledge) des Prozessors, der über eine Leitung mit dem Interruptcontroller verbunden ist, signalisiert die CPU diesem in einem ersten Zyklus, dass der Interrupt angenommen wurde. Der INTA-Ausgang ist nullaktiv und flankengesteuert. Im zweiten Signalzyklus der INTA-Leitung legt der Interruptcontroller die Interruptnummer des an ihm anliegenden Interrupts auf den Datenbus. Dieser Wert wird vom Prozessor über den Datenbus aufgenommen. [8, 2-22 - 2-29]

Um den Speicherort der Fortsetzadresse für das Interruptprogramm zu erhalten, multipliziert der 8086 die aufgenommene hexadezimale Interruptnummer mit einer dezimalen Vier. Die Multiplikation mit 4 hängt damit zusammen, dass für die Fortsetzadresse Inhalte für das CS- und IP-Register geholt werden. Bei 1 Byte Speicherzellen und 16-Bit Registerbreite sind somit 4 Speicherzellen notwendig. Aus der so entstandenen Speicheradresse und weiteren drei direkt darüberliegenden holt sich der Prozessor dann den Inhalt für das IP-Register aus den unteren beiden Speicherzellen und den Inhalt des CS-Registers aus den oberen beiden Speicherzellen. Diese Speicherzellen liegen in der Interruptvektortabelle. Durch eine Offsetaddition der Register ergibt sich die Speicheradresse für das Interruptunterprogamm, mit welchem der Prozessor an dieser Stelle fortfährt. [8, 2-22 - 2-29]

Am Ende einer jeden Interrupt-Service-Routine muss ein IRET-Befehl (Interrupt-Return) stehen, ansonsten ist es dem Prozessor nach der Abarbeitung der ISR nicht möglich, mit dem zuvor unterbrochenen Programm fortzufahren. Mit der Abarbeitung des IRET-Befehls holt sich der Prozessor die auf dem Stack abgelegte Rücksprungadresse wieder und lädt die Register

mit den entsprechenden Inhalten. Klassischerweise ist Nesting beim 8086 verboten. Der NMI hat außerdem immer eine höhere Priorität als der INTR. [8, 2-22 - 2-29] [9]

Abbildung 5 zeigt den zeitlichen Verlauf der Prozessorsignale bei der Annahme eines Interrupts.

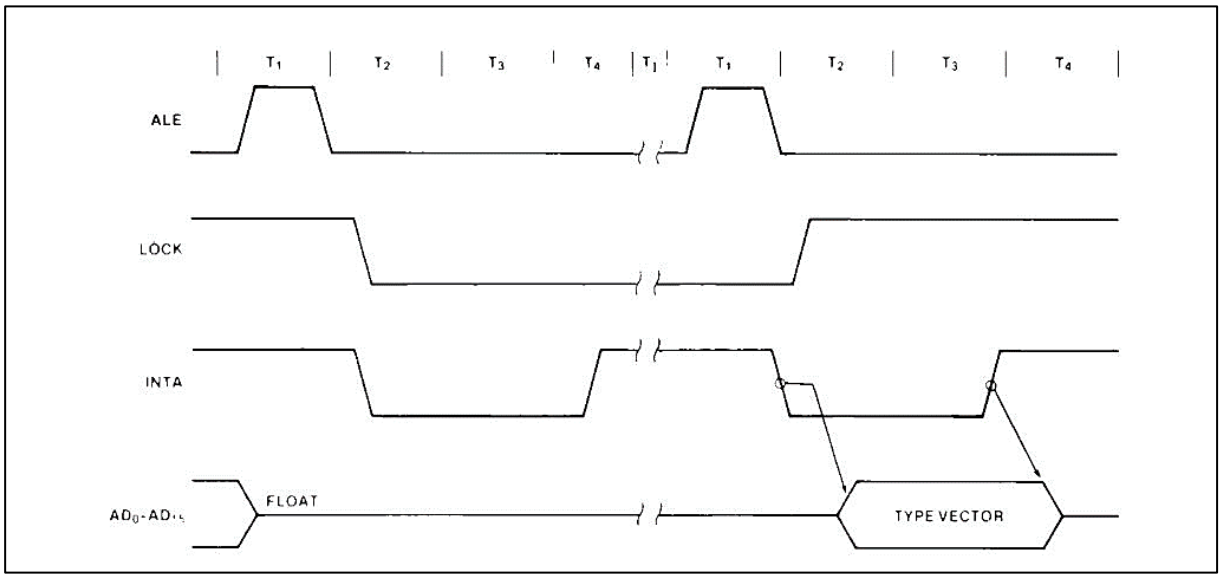


Abbildung 5 – Zeitlicher Verlauf der Prozessorsignale bei der Interruptannahme eines Intel 8086 Mikroprozessors [10, S. 11]

6.3.1 Der Interruptcontroller

Der Interruptcontroller organisiert die extern eingehenden Interrupts und kontrolliert die INTR-Leitung. Er entscheidet beispielsweise über die Priorität der eingehenden Interrupts und reiht diese in eine Warteschlange ein. Beim 8086 wird für gewöhnlich ein Intel 8259A programmierbarer Interruptcontroller verwendet. Dieser besitzt 8 Interrupteingänge ist jedoch mit weiteren 8259A bis auf 256 Eingänge kaskadierbar, um die volle Interruptanzahl des 8086 auszuschöpfen. [8, 2-22 - 2-29, A-135 ff.]

6.3.2 Die Interruptvektortabelle

Die Interruptvektortabelle enthält alle Fortsetzadressen für Interruptprogramme und liegt im unteren ersten Kilobyte des RAMs. Bei der Initialisierung des RAMs wird diese dort platziert. *Abbildung 6* zeigt schematisch die Belegung der Interruptvektortabelle im RAM.

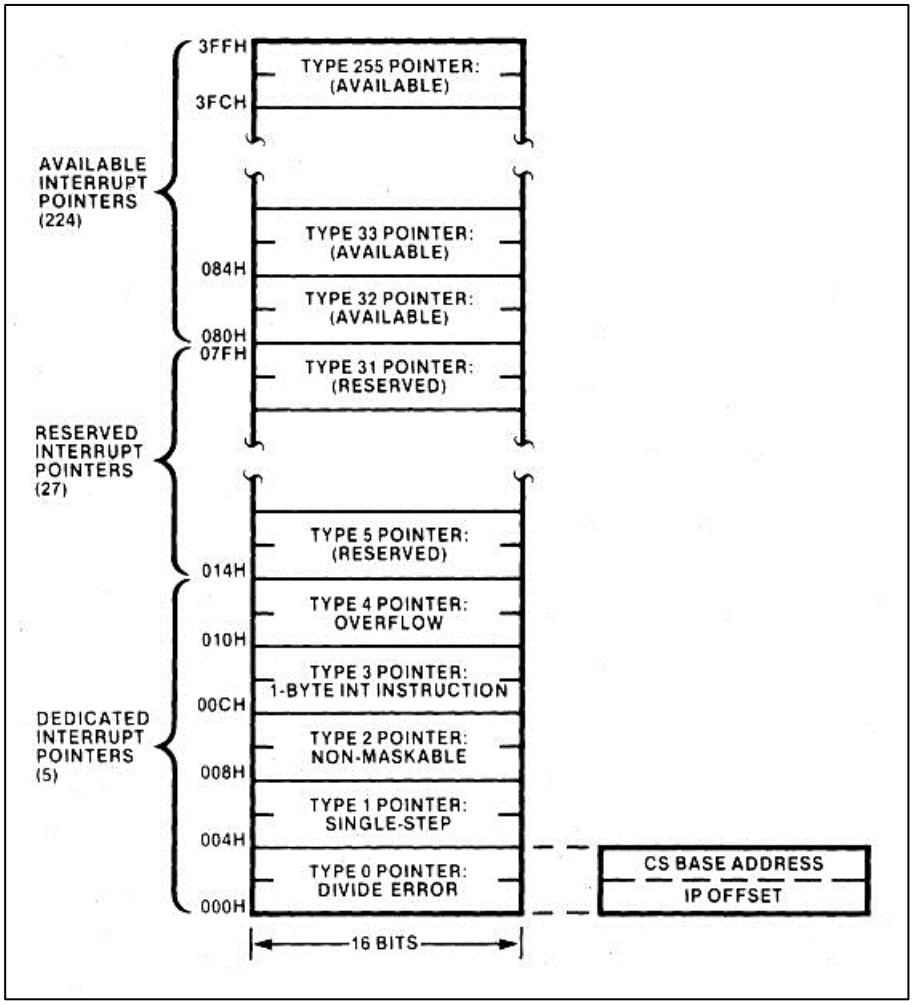


Abbildung 6 – Schematische Darstellung der Belegung der Interruptvektortabelle im RAM bei Interruptverarbeitung mit einem Intel 8086 Mikroprozessor [8, S. 2-25]

Zu sehen ist, dass dort bereits einige Einträge vorhanden sind. Ganz am unteren Ende der Tabelle befinden sich die Standard-Interruptroutinen des 8086, wie z.B. die Behandlung von Divisionen durch Null, die Verarbeitung des NMI und Registerüberläufe. Darüber schließt sich ein Bereich an, der freigehalten werden sollte, um die Kompatibilität mit zukünftigen Intel Prozessoren dieser Generation sicherzustellen. Der Interrupt 32 ist der erste Interrupt der extern verwendet werden kann. Die Interruptvektortabelle wird beim Start eines Programms durch das Programm selbst mit den notwendigen Werten beschrieben. [8, 2-22 - 2-29]

6.4 Digital – Simulator für digitale Schaltkreise

Die verwendete Software mit dem Namen „Digital“ ist ein Simulator für digitale Schaltkreise und kostenfrei via GPL (General Public Licence) erhältlich. Die Anwendung ist auf den Betriebssystemen Linux, Windows und MacOS lauffähig und kann ohne Installation verwendet

werden. Entwickler ist der Hochschulprofessor Helmut Neemann. Der Simulator kann unter folgendem Link von der Entwicklerplattform GitHub heruntergeladen werden. [1]

https://github.com/hneemann/Digital

Die Software verfügt über eine detaillierte Dokumentation in verschiedenen Sprachen und über zahlreiche Funktionen. Gleichzeitig ist sie für die studentische Verwendung entwickelt worden und daher, leicht zu bedienen. Die Hilfe-Funktion des Simulators platziert die Informationen aus der Dokumentation an der jeweils richtigen Stelle und verfügt teilweise auch über zusätzliche Informationen. Sowohl die grundlegenden logischen Gatter als auch komplexe Schaltnetze und Schaltwerke, wie Zähler und arithmetische Schaltungen bis hin zu Speichern, sind dort als Bausteine modelliert und verfügbar. Zusätzlich zu einer breiten Auswahl an I/O-Bausteinen stellt der Entwickler bereits eine große Anzahl von Beispielschaltungen, wie z.B. einen voll funktionsfähigen Prozessor zur Verfügung. Auch Nachbildungen von echten Chips sind verfügbar. Der Simulator verfügt über einen Entwicklungsmodus und einen Simulationsmodus. Im Entwicklungsmodus können Schaltungen mithilfe der Bausteine und Leitungen entworfen werden. Im Simulationsmodus können diese Schaltungen dann simuliert werden. Dazu werden Eingänge gesetzt, und die Software berechnet entsprechende Ausgänge. Dabei sind beliebige Komplexitäten bis hin zu Mikrocomputern möglich. [11]

Im Folgenden einige Funktionen des Simulators.

- Automatische Generierung des booleschen Ausdrucks zu einer angelegten Schaltung
- Automatische Synthese einer Schaltung, die einen eingegebenen booleschen Ausdruck erfüllt
- Automatische Minimierung boolescher Ausdrücke
- Simulation von endlichen Automaten in Zustandsdiagrammen
- Erstellen von generischen Schaltungen
- Skriptgesteuertes Testen
- Schaltungsexport in Dateiformate zur Chipfertigung
- Schaltungsexport zu VHDL (**V**ery High Speed **H**ardware **D**escription **L**anguage)

- Import benutzerdefinierter Schaltungssymbole
- Einbetten von Schaltungen

[1] [11]

Zahlreiche Visualisierungsbausteine, wie LEDs (Leuchtkreise), Taster mit LEDs, Ein- und Ausgänge mit angezeigtem Wert und Messwertgraphen sowie das Platzieren von Textfeldern und Rahmen, lassen eine effiziente optische Aufbereitung von Schaltungen zu. Die farbliche Veränderung der Leitungen abhängig vom Zustand (high oder low) ist ebenfalls vorteilhaft. [1] [11]

Weitere Informationen zur Software können der Website unter obenstehendem Link oder der Dokumentation entnommen werden.

7 Entwicklung der Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors

7.1 Erste Testschaltungen

Zu Beginn sind einige Testschaltungen entstanden, um die Fähigkeiten des Programms auszutesten und ein Gefühl für den Schaltungsentwurf zu bekommen. Dabei war von vorneherein klar, dass das Arbeiten mit Teilschaltungen und das Ineinandereinbetten dieser bis in die oberste Ebene zu einer Hauptschaltung für die Visualisierung sehr vorteilhaft sein kann. So enthalten die Teilschaltungen alle notwendigen funktionalen Elemente für die Interruptverarbeitung, verwirren aber durch ihre Komplexität nicht den Benutzer der Simulation. In der obersten Ebene werden dem Anwender dann nur noch eingebettete Teilschaltungen als Bausteine und für das Verständnis Interruptverarbeitung relevante Elemente gezeigt. Die Teilschaltungen repräsentieren dabei jeweils einen realen Teil eines Mikrocomputers (z.B. Prozessor oder Interruptcontroller).

Abbildung 7 zeigt eine erste Testschaltung für den Interruptcontroller. In dieser wird lediglich durch das eingehende INTA-Signal eine zuvor eingegebene Interruptnummer über einen Leitungstreiber auf einen Ausgang ausgegeben. Gleichzeitig wird ein INTR-Signal ausgegeben. Die Arbeit mit Leitungstreibern ist für das gesamte Projekt relevant, da gerade auf dem Datenbus mehrere Teilnehmer lesen und schreiben. Eine abgestimmte Schaltung von

Leitungstreibern muss dabei Kurzschlüsse verhindern, indem immer nur ein Leitungstreiber angesteuert wird. Ungesteuerte Leitungstreiber sind an ihrem Ausgang hochohmig (Z) geschaltet, angesteuerte schalten den Eingang auf den Ausgang durch. Daher sollten Busleitungen in der Regel immer mit Leitungstreibern abgeschlossen werden.

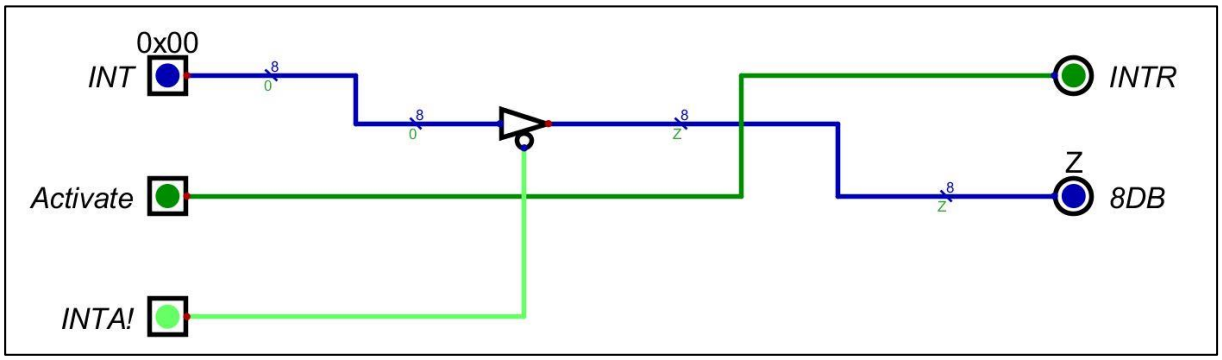


Abbildung 7 – Erste Testschaltung eines Interruptcontrollers – Die Interruptnummer wird über einen Leitungstreiber auf den Datenbus geschaltet

Abbildung 8 zeigt die Schaltung aus *Abbildung 7* als eingebettete Schaltung in einer neuen Schaltung. Das INTA-Signal wird durch eine einfache Negierung des INTR-Signals erzeugt und somit auf einfache Weise die Reaktion eines Prozessors simuliert.

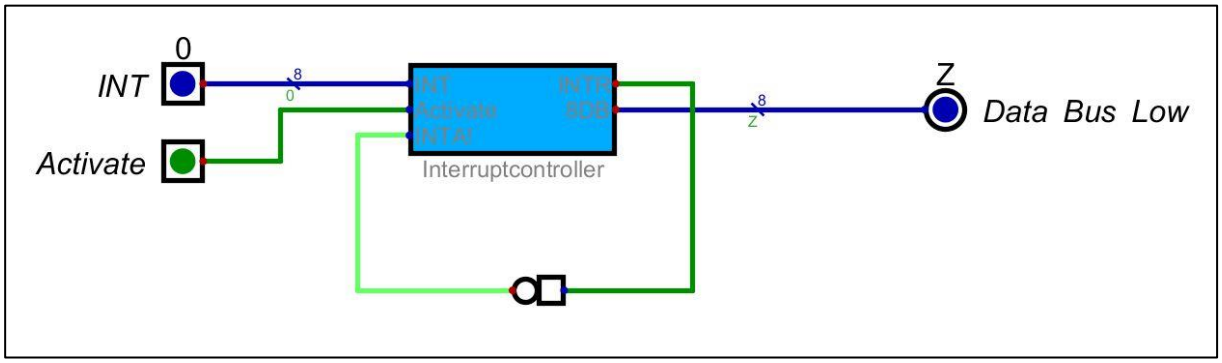


Abbildung 8 – Erste Testschaltung für die Übergabe der Interruptnummer durch den Interruptcontroller als eingebettete Schaltung

7.2 Prototyp der Hardware-Interruptverarbeitung

Im Vergleich zu den ersten Testschaltungen handelt es sich bei diesem Prototypen bereits um ein komplettes Modell der Hardware-Interruptverarbeitung.

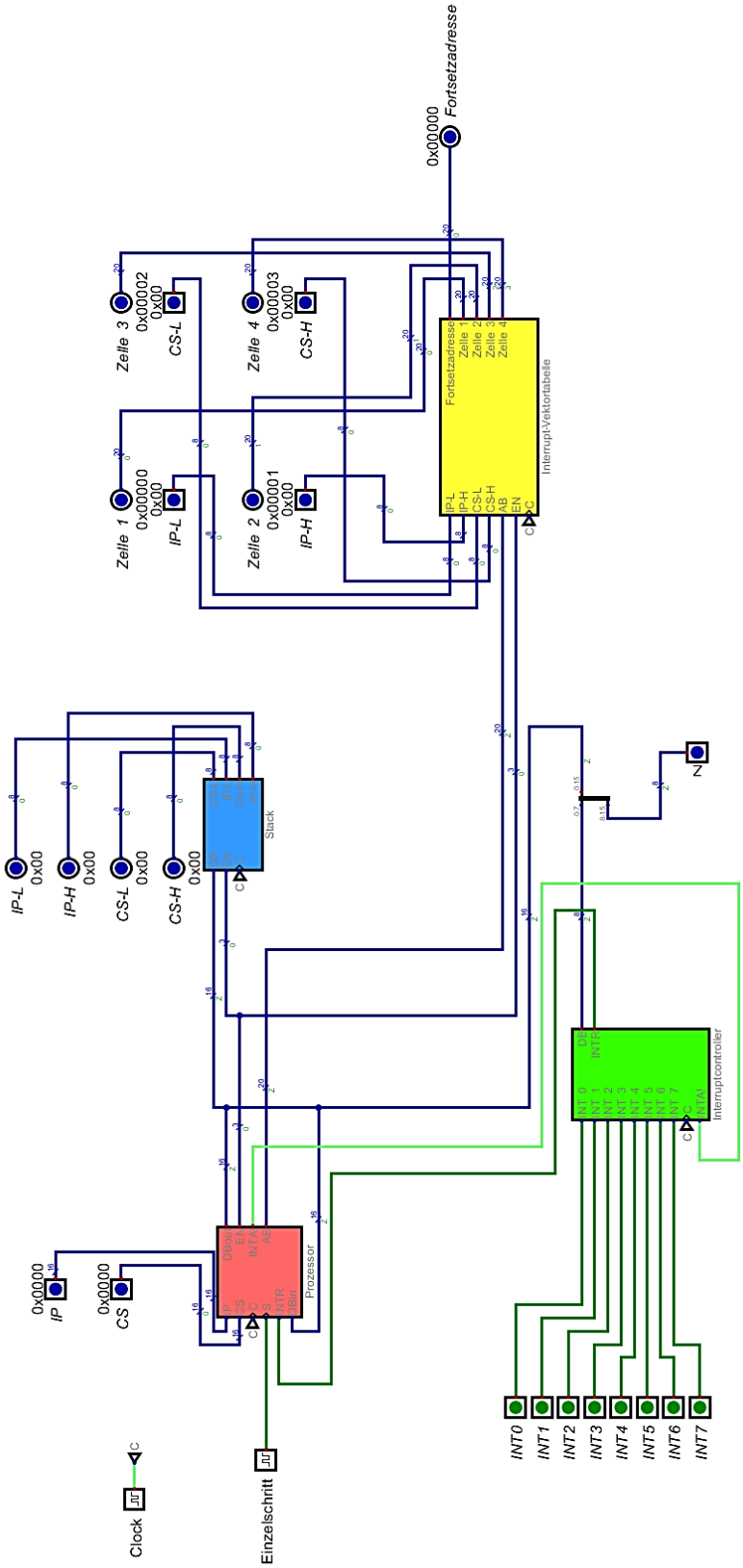


Abbildung 9 – Prototyp der Visualisierung der Hardware-Interruptverarbeitung – Simulatorschaltung für Digital

Das Konzept hinter dem Prototypen ist folgendes. Der Interruptcontroller nimmt nach dem Vorbild des Intel 8259A die Interrupts INT0 bis INT7 an und verwaltet sie. Außerdem hat er einen Eingang für das INTA-Signal, einen Ausgang für das INTR-Signal und ist an den Datenbus angeschlossen. Sobald ein Interrupt aktiviert wird, sendet der Interruptcontroller durch das INTR-Signal eine Interruptanfrage an den Prozessor. Dieser legt daraufhin die Register CS und IP auf den Stack. Der Stack in *Abbildung 9* blau dargestellt zeigt diese Register daraufhin an. Ist dieser Schritt abgeschlossen, signalisiert der Prozessor dem Interruptcontroller durch das INTA-Signal, dass dieser nun die Interruptnummer auf den Datenbus legen kann, was dieser dann auch tut. Mithilfe der Interruptnummer berechnet der Prozessor dann die Adresse in der Interruptvektortabelle an der die Fortsetzadresse für das Interruptprogramm in Form der Registerinhalte für IP und CS liegt. Diese Adresse wird vom Prozessor über einen Adressbus an den gelben Baustein übergeben, der die Interruptvektortabelle darstellt. Durch die Ausgänge an diesem Baustein wird dann angezeigt in welchen Speicherzellen die Registerinhalte für die Fortsetzadresse liegen. Mit diesem Schritt ist die Interruptverarbeitung bei diesem Prototypen beendet.

Bei diesem Prototypen ist die Interruptverarbeitung wesentlich früher beendet als in der Realität, nämlich bereits dann, wenn die Fortsetzadresse für das Interruptprogramm bekannt geworden ist. Außerdem sind der Stack und die Interruptvektortabelle eigenständige Bausteine, obwohl sie in Realität nur Bereiche im RAM bezeichnen. Dieser Aufbau wurde gewählt, um die Visualisierung von Vektortabelle und Stack zu verbessern und gleichzeitig die Verwendung eines RAM-Bausteins zu vermeiden.

Abbildung 10 zeigt die Schaltung des in *Abbildung 9* rot dargestellten Prozessors. Bereits bei diesem Prototyp war ein Durchlauf in Einzelschritttaktung möglich. Dieser steuert gleichzeitig über eine Steuerleitung die Leitungstreiber für Adress- und Datenbus in allen Bausteinen der Schaltung. Durch einen Eingang und einen nachgeschalteten Zähler, dem wiederum ein Decoder nachgeschaltet ist, wird mit jeder Betätigung des Eingangs eine andere Steuerleitung aktiviert. Diese Steuerleitungen bewirken dann, je nach dem in welchem Schritt man sich befindet, hauptsächlich über Leitungstreiber den gewünschten Effekt innerhalb der Hardware-Interruptverarbeitung. Das codierte Steuersignal wird über eine 3-Bit Leitung an alle Bausteine verteilt und lokal decodiert. Ausgenommen ist der Interruptcontroller, da der

Prozessor über INTA und INTR mit diesem kommuniziert. Bei der Planung war nicht klar, ob sich die Einzelschritttaktung einfach würde realisieren lassen. Bereits beim Prototyp stellte sich jedoch heraus, dass diese sogar große Vorteile für die Funktionalität hat und die Modellierung einfacher macht.

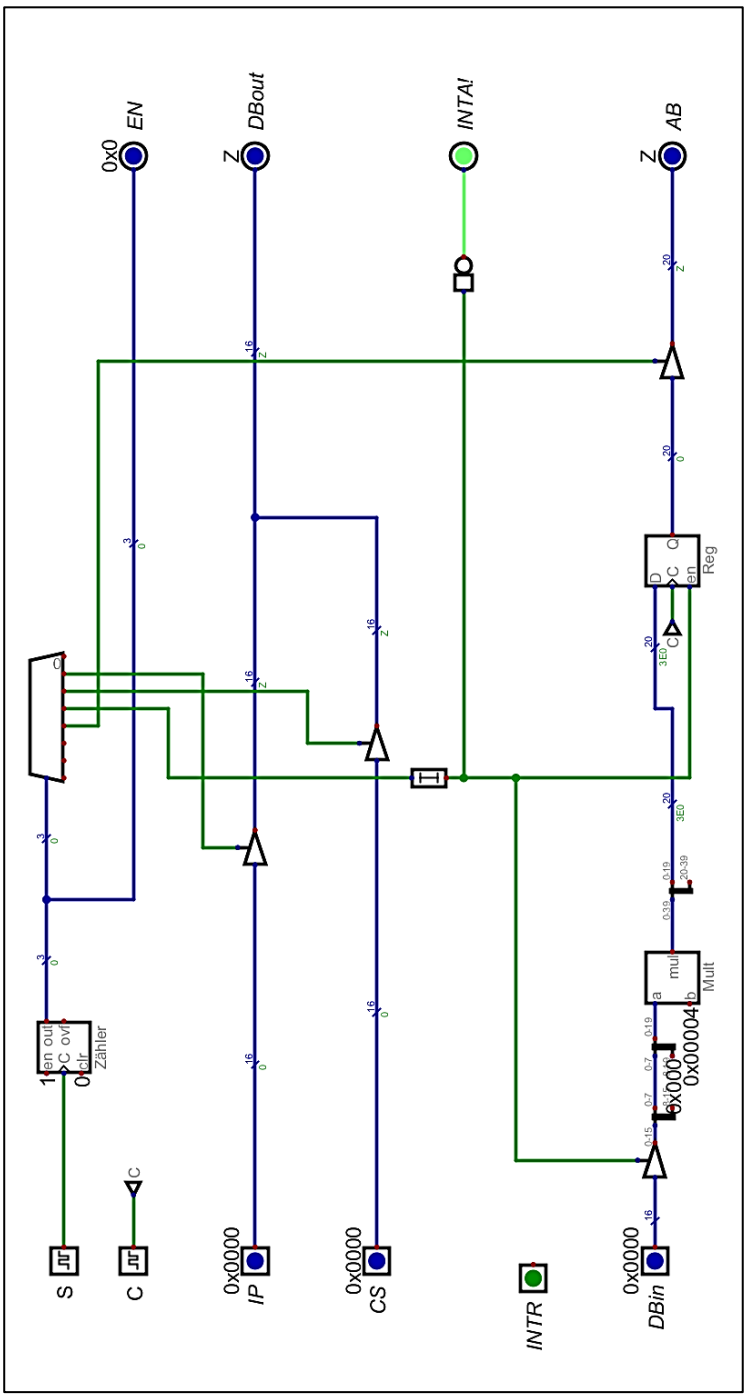


Abbildung 10 – Prozessorschaltung des Prototyp der Visualisierung der Hardware-Interruptverarbeitung – Simulatorschaltung für Digital

8 Endversion der Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors

Abbildung 11 zeigt die Hauptschaltung der Endversion der Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors. Grundlegender Unterschied der Endversion zum Prototyp ist der hinzugekommene RAM-Baustein, in dem sich realitätsgetreu der Stack und die Interruptvektortabelle befinden. Um unnötige Verwirrung durch viele Leitungen zu vermeiden, ist viel mit Tunneln umgesetzt worden. So ist jede Leitung, die nur durch visualisierungstechnische Elemente notwendig wurde und nicht wichtiger Bestandteil eines Mikrocomputers ist, durch einen Tunnel ersetzt. Ausführliche Beschreibungen und Hinweise erleichtern die Bedienung der Simulation und fördern das Verständnis des Gezeigten.

Des Weiteren simuliert man die Hardware-Interruptverarbeitung in der Endversion bis zu dem Punkt, an dem in der Realität das Interruptprogramm starten würde. Somit wird gegenüber dem Prototyp auch das Laden der Fortsetzadresse in den Prozessor noch mitsimuliert. Das Rückholen der Rücksprungadresse aus dem Stack nach der Abarbeitung des Interruptprogramms lässt man der Einfachheit halber weg. Ohnehin läuft der Hauptteil der Verarbeitung vor dem Interruptprogramm ab.

8.1 Der Interruptcontroller

Die äußere Erscheinungsform des Interruptcontrollers hat sich seit dem Prototyp kaum verändert. Die innere Verschaltung des Interruptcontrollers ist in *Abbildung 12* gezeigt.

Zum Auslösen der Interrupts INT0 bis INT7 steht jeweils ein Taster zur Verfügung. Der daraufhin entstehende Puls wird von einem Encoder im Interruptcontroller codiert und anschließend von einem Register gehalten. Gleichzeit erfolgt das Setzen des INTR-Signals über ein RS-Flipflop. Wenn der Prozessor durch eine Null auf der INTA-Leitung den Interrupt annimmt, wird dieses RS-Flipflop rückgesetzt, sodass das INTR-Signal verschwindet. Außerdem wird ein Leitungstreiber angesteuert, welcher die im Register gehaltene Interruptnummer auf den 8-Bit Datenbusausgang gibt.

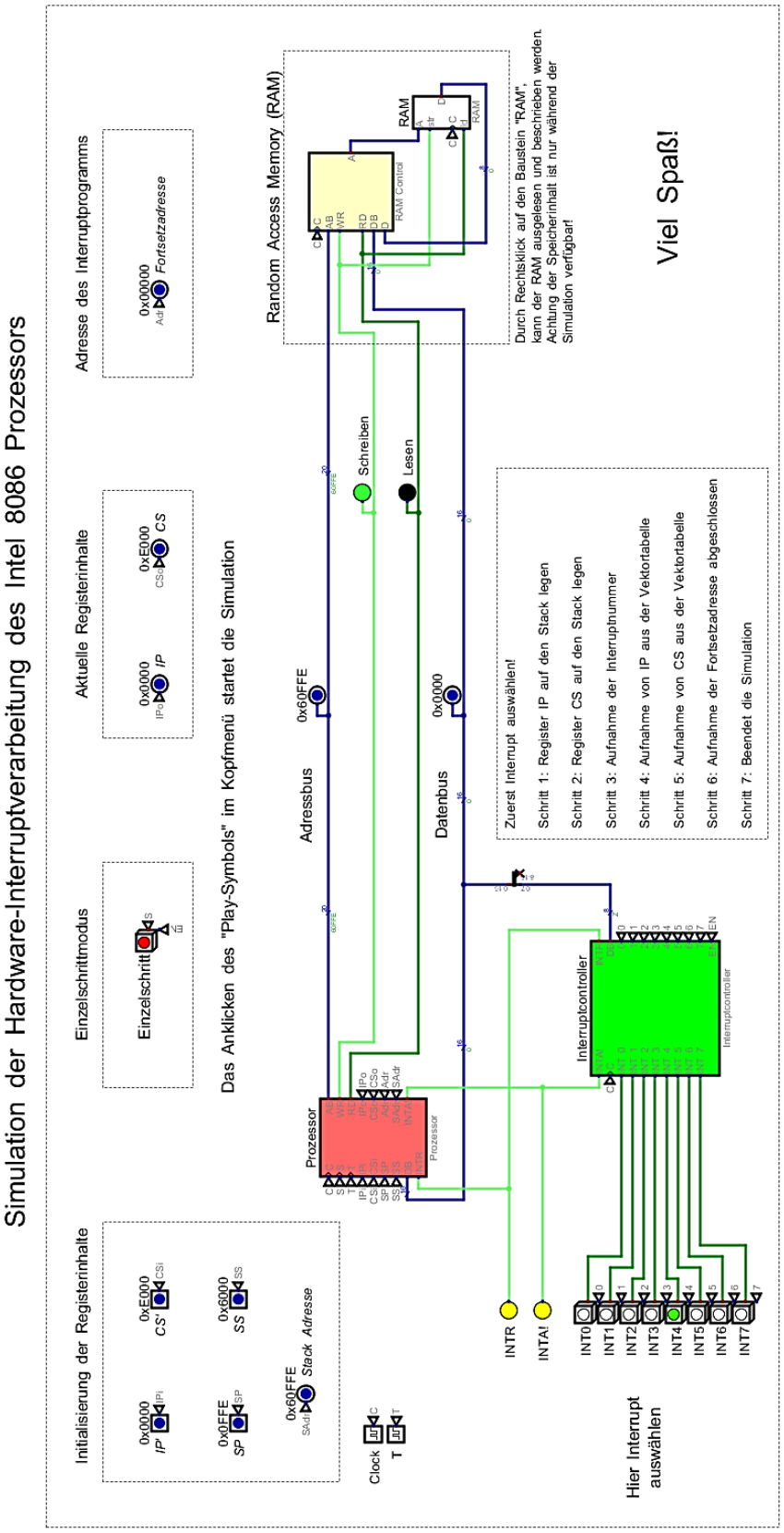


Abbildung 11 – Endversion der Hauptschaltung der Simulation der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors

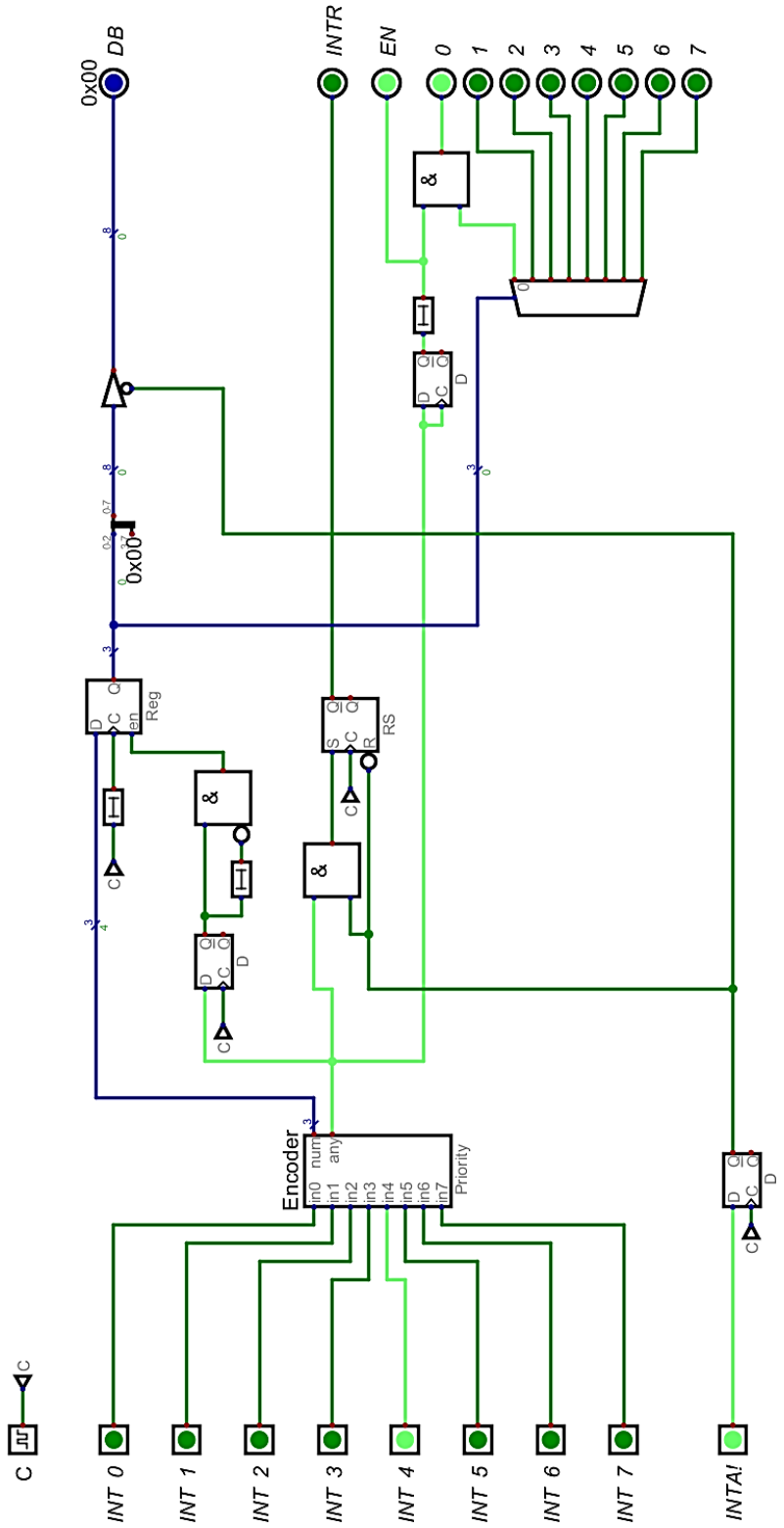


Abbildung 12 – Schaltung des Interruptcontrollers in der Endversion der Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors

8.2 Der Arbeitsspeicher

Beim Überdenken des Prototypen hat sich herausgestellt, dass eine Visualisierung mittels RAM doch besser ist als die vorherige Lösung. Dies lässt sich primär durch die Nähe zur Realität argumentieren. Außerdem bietet der RAM wesentlich mehr Möglichkeiten, so können der Stack und die Interruptvektortabelle nun beliebig ausgelesen und beschrieben werden. Verwendet ist ein Standard RAM-Baustein des Simulators, der sich im Simulationsmodus durch einen Klick auf diesen auslesen und beschreiben lässt. Dies passiert mittels einer Matrix, die jede Speicherzelle des RAMs abbildet. Die Größe des RAMs richtet sich dabei nach der Größe des angeschlossenen Adressbusses. Die Größe des angeschlossenen Datenbusses gibt die Größe der Speicherzellen vor. Hierbei ergibt sich ein Problem. Durch den 16-Bit Datenbus wären die Speicherzellen des RAM doppelt so groß wie bei Mikrocomputern üblich, da diese üblicherweise einen Zwei-Block-RAM besitzen. In dieser Visualisierung soll es der Übersicht wegen allerdings nur einen RAM-Baustein geben. Um dieses Problem zu umgehen, existiert eine weitere eingebettete Teilschaltung mit dem Namen „RAM Control“. Diese konvertiert den eingehenden 16-Bit Datenbus auf einen 8-Bit Datenbus und schreibt die Inhalte in 2 Zyklen in den RAM. Das Lesen erfolgt andersherum auf dieselbe Weise.

Der RAM selbst wird durch zwei Steuerleitungen (lesen und schreiben) gesteuert und besitzt selbstverständlich einen Anschluss für den Adress- und Datenbus.

8.3 Der Prozessor

Abbildung 13 zeigt die Schaltung, welche in der Visualisierung der Hardware-Interruptverarbeitung die Funktion des Prozessors übernimmt. Auch in der Endversion wird die Einzelschritttaktung über einen Zähler mit nachgeschaltetem Decoder eingebunden. Diese steuert auch weiterhin den Hauptteil der Leitungstreiber mit dem Unterschied, dass diese Steuersignal nun nur noch im Prozessor existieren, da die Kommunikation mit dem RAM über andere Steuerleitungen erfolgt. Prinzipiell bearbeitet der Prozessor dieselben Aufgaben wie schon der Prozessor im Prototyp allerdings mit höherem Funktionsumfang und daher höherer Komplexität. Aufgrund der hinzugekommenen Komplexität sind einige Funktionen des Prozessors in weitere universaleinsetzbare Teilschaltungen ausgelagert worden, um die Übersicht zu behalten.

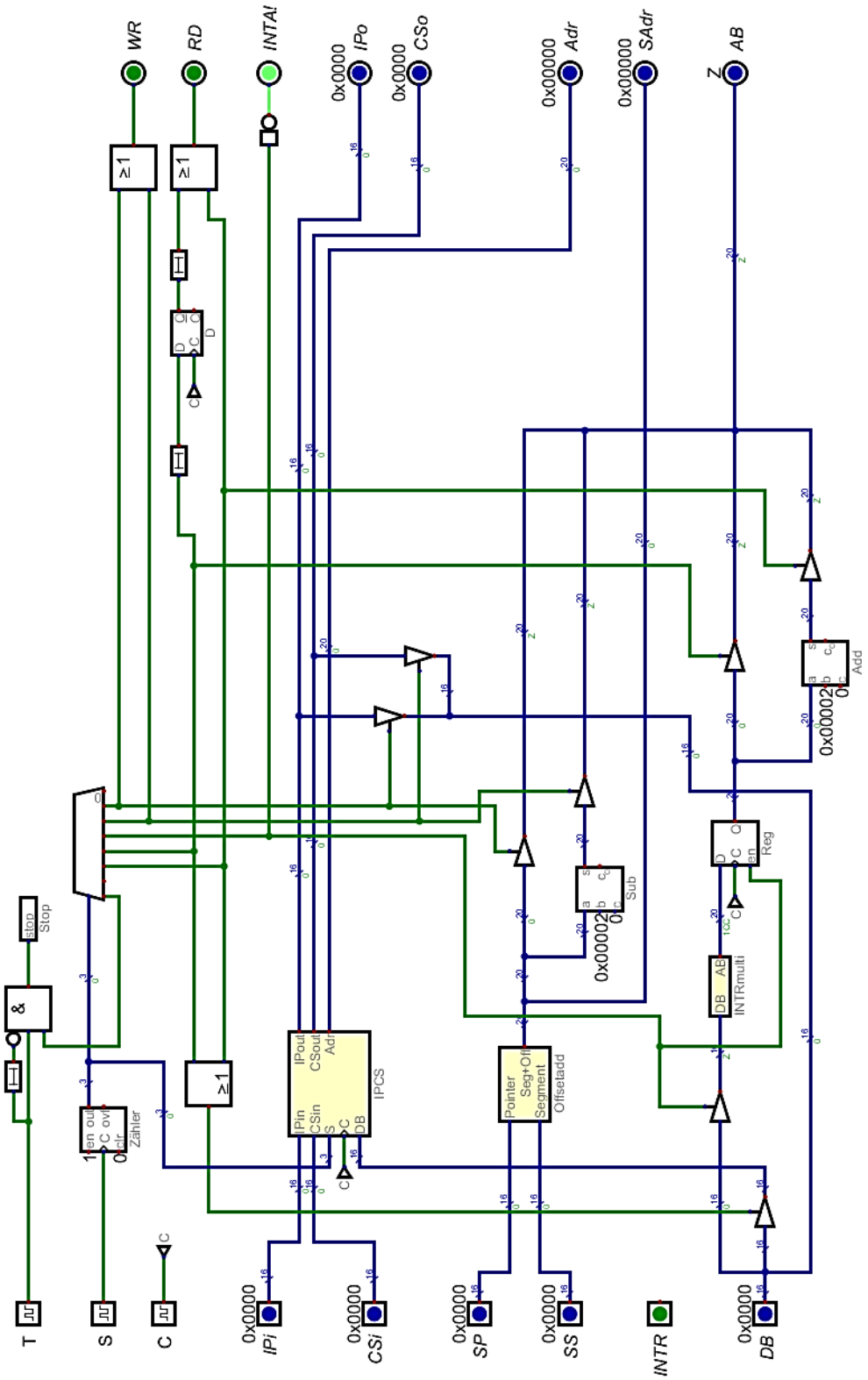


Abbildung 13 - Schaltung des Prozessors in der Endversion der Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors

Die Teilschaltung INTRmulti multipliziert ihren 16-Bit Eingang mit 4 und stellt am Ausgang einen 20-Bit Wert bereit.

Die Teilschaltung Offsetadd führt eine Offsetaddition des Segments und des Pointers durch, welche an den Eingängen anliegen. Ausgegeben wird eine entsprechende 20-Bit Adresse.

Die Teilschaltung IPCS koordiniert die die Anzeigen für die Registerinhalte von IP und CS innerhalb des Prozessors.

Ein spezieller Schaltungsteil innerhalb der Prozessorschaltung sorgt dafür, dass beim siebten Einzelschritt der Simulation, wenn die Verarbeitung des Interrupts bereits abgeschlossen ist, der Simulationsmodus von Digital automatisch beendet wird.

8.4 Schaltungstakt

Die gesamte Schaltung besitzt zwei Taktgeneratoren. Der erste ist der standardmäßige Schaltungstakt, welcher in der Realität von einem Quarz erzeugt wird und die Flanken für die taktflankengesteuerten Schaltwerke liefert. Dieser Takt darf nicht zu gering gewählt werden, da ansonsten Lauftzeiterscheinungen bezüglich des Einzelschritttaktes auftreten können. Das liegt daran, dass Digital auch Gatterlaufzeiten simuliert. Der Schaltungstakt ist daher mit 100 Hz gewählt. Eine höhere Frequenz wäre in der Simulation zwar ohne Probleme möglich, da im Gegensatz zur Realität hier keine Probleme wie Überhitzung durch zu hohe Verlustenergie oder Wanderwellenerscheinungen auftreten, jedoch ist die gewählte für eine so einfache Schaltung völlig ausreichend.

Der zweite Taktgenerator liefert einen sehr niederfrequenten Zusatztakt, der zum automatischen Beenden der Simulation beim letzten Einzelschritt benötigt wird.

8.5 Vorbereitung der Simulation

Um in Digital in den Simulationsmodus zu wechseln wird, auf das „Play-Symbol“ in der Kopfleiste des Programms geklickt. Durch den Knopf mit dem „Stopp-Symbol“ weiter rechts kann die Simulation zu jedem Zeitpunkt gestoppt werden, und das Programm kehrt zurück in den Bearbeitungsmodus.

Sobald der Simulationsmodus gestartet wurde, ist es möglich, eine Hardware-Interruptverarbeitung im Einzelschritt durchzugehen. Zuvor können optional noch einige Voreinstellungen vorgenommen werden.

1. Es ist möglich, die initialisierten Inhalte der Register IP, CS, SP und SS zu verändern. Dies ist im Feld „Initialisierung der Registerinhalte“ links oben in der Simulation möglich. Natürlich können auch einfach die voreingestellten Inhalte beibehalten werden. Außerdem wird unterhalb des Stapelsegments (SS – Stack Segment) und des Stapelzeigers (SP – Stack Pointer) direkt die sich daraus ergebende Stapeladresse (Stack Adresse) angezeigt. Um die Inhalte der Register zu verändern, ist lediglich ein Klick auf die Eingangssymbole notwendig. Daraufhin erscheint ein Fenster, in dem der Wert angepasst werden kann.
2. Weiterhin können die Speicherzellen des RAM voreingestellt werden. Dies ist notwendig, um die Interruptvektortabelle zu initialisieren. Der RAM hat entsprechend des Adressbusses eine Größe von 1 MiB (1 Mega Binary Byte) und wird beim Wechseln in den Simulationsmodus programmseitig mit 00h je Speicherzelle beschrieben. Um den RAM manuell zu beschreiben, klickt man mit der Maus im Feld „Random Access Memory (RAM)“ auf den unteren rechten Baustein, der die Überschrift RAM trägt. Daraufhin öffnet sich ein neues Fenster mit einer Matrix, in der man dann jede beliebige Speicherzelle beschreiben kann.

8.6 Ablauf der Simulation

Die Simulation der Hardware-Interruptverarbeitung erfolgt im Einzelschrittmodus, sodass alle Verarbeitungsschritte in Ruhe nachvollzogen werden können.

Zuerst muss ein Hardware-Interrupt durchgeführt werden. Dazu stehen unten Links in der Hauptschaltung 8 Taster mit den Bezeichnungen INT0 bis INT7 zur Verfügung. Durch Drücken eines Tasters wird der jeweilige Interrupt ausgelöst und vom Interruptcontroller registriert. Der Taster des Interrupts leuchtet danach grün.

Achtung: Werden mehrere Taster nacheinander gedrückt, so ist immer der Interrupt aktiv der zuletzt ausgelöst wurde. Die Taster, die davor gedrückt wurden, haben keine Auswirkung mehr auf die Interruptverarbeitung.

Sobald ein Interrupt ausgelöst wurde, kann die Verarbeitung im Einzelschritt durchlaufen werden. Um einen Schritt weiterzugehen, wird der Taster „Einzelschritt“ im Feld „Einzelschrittmodus“ am oberen Rand der Schaltung verwendet. Dieser Taster leuchtet rot, sobald ein Interrupt ausgelöst wurde und indiziert dadurch, dass die Simulation nun bereit ist im Einzelschritt durchlaufen zu werden. Durch jede Betätigung des Tasters geht die Simulation einen Schritt weiter in der Hardware-Interruptverarbeitung.

Folgende Schritte werden nacheinander ausgeführt.

Schritt 1 Register IP auf den Stack legen

In diesem Schritt schreibt der Prozessor den Inhalt des Registers IP mit seinem initialisierten Wert auf den Stapel im RAM.

Schritt 2 Register CS auf den Stack legen

In diesem Schritt schreibt der Prozessor den Inhalt des Registers CS mit seinem initialisierten Wert auf den Stapel im RAM.

Schritt 3 Aufnahme der Interruptnummer

In diesem Schritt nimmt der Prozessor die vom Interruptcontroller auf den Datenbus gelegte Interruptnummer entsprechend des ausgelösten Interrupts auf.

Schritt 4 Aufnahme von IP aus der Vektortabelle

In diesem Schritt liest der Prozessor entsprechend der aufgenommenen Interruptnummer den neuen Wert für das Register IP aus der Interruptvektortabelle im RAM.

Schritt 5 Aufnahme von CS aus der Vektortabelle

In diesem Schritt liest der Prozessor entsprechend der aufgenommenen Interruptnummer den neuen Wert für das Register CS aus der Interruptvektortabelle im RAM.

Schritt 6 Aufnahme der Fortsetzadresse abgeschlossen

Nach Schritt 5 ist durch die Aufnahme der neuen Werte für IP und CS die Adresse für das Interruptprogramm bekannt. Die Aufnahme der Fortsetzadresse ist somit abgeschlossen, und der Prozessor geht in diesem Schritt zurück in den Ruhezustand über. Die weitere Hardware-Interruptverarbeitung ist ab diesem Punkt nicht weiter simuliert, da nun ein zuvor geschriebenes Interruptprogramm abgearbeitet würde. Auch das Zurückholen der ursprünglichen Inhalte des IP und CS Registers vom Stapel wird nicht mehr simuliert. Der simulierte Teil der Hardware-Interruptverarbeitung ist daher an dieser Stelle beendet.

Schritt 7 Beendet die Simulation

Ein siebtes Betätigen des Einzelschritttasters beendet die Simulation vollständig. Das Programm kehrt dabei aus dem Simulationsmodus zurück in den Bearbeitungsmodus.

Diese Einzelschritte sind in ihrer Kurzform auch in der Schaltung wieder zu finden.

8.7 Optische Indikatoren

Um eine bessere Visualisierung zu erreichen, enthält die Schaltung bzw. die Simulation einige optische Indikatoren.

1. Im Feld „Aktuelle Registerinhalte“ können die Inhalte der Register IP und CS zu jedem Zeitpunkt während der Simulation beobachtet werden.
2. Im Feld „Adresse des Interruptprogramms“ kann am Ende der Simulation die Fortsetzadresse für das Interruptprogramm, ermittelt aus den Registern CS und IP, abgelesen werden.
3. Grundsätzlich gilt für Leitungen in Digital während der Simulation Folgendes.
    - Eine 1 Bit Leitung ist hellgrün, wenn ihr Zustand „high“ ist. Sie erscheint dunkelgrün, wenn ihr Zustand „low“ ist.

- Ein Bus wird ebenfalls als eine Leitung dargestellt, jedoch wird an dieser die Anzahl der Adern angezeigt. Der Zustand bzw. Wert eines Buses während der Simulation kann durch grüne Zeichen an diesem festgestellt werden.

4. Der Zustand des Adress- und des Datenbusses kann zusätzlich durch jeweils eine Anzeige beobachtet werden. Der Wert der Busse wird dabei hexadezimal angegeben, wobei der Buchstabe „Z“ für hochohmig steht.
5. Die Zustände der Leitungen für das Lesen und Schreiben von oder in den RAM sowie die Zustände der Leitungen INTR (Interrupt) und INTA! (Interrupt Acknowledge) werden zusätzlich durch Leuchtkreise angezeigt. Ist der Kreis farbig, so ist die Leitung „high“, ist er schwarz, so ist die Leitung „low“.

8.8 Bedienungsanleitung

Eine Bedienungsanleitung für die Simulation der Hardware-Interruptverarbeitung des Intel 8086 Prozessors ist in *Anhang 4 – Bedienungsanleitung für die Simulation der Hardware-Interruptverarbeitung des Intel 8086 Prozessors* enthalten.

9 Testung

Zur Qualitätssicherung und Funktionskontrolle sind während der Entwicklung diverse Tests durchgeführt worden. Dabei wird grundsätzlich nach jedem Entwicklungsschritt ein Test mit dem Kunden durchgeführt.

Außerdem werden nach jedem Entwicklungsschritt die einzelnen Komponenten sowie die Komponenten im Verbund getestet. Dabei wird ermittelt, ob bestimmte Eingaben die gewünschten Ergebnisse erzielen. So werden bei jeder Teilschaltung die Auswirkungen der relevanten Signale untersucht. Bei Busleitungen werden grundsätzlich stichprobenartige Tests durchgeführt. Sind diese positiv, so wird der gesamte Bus als intakt angesehen.

Aufgrund der Eigenschaft von Digital, Kurzschlüsse auf Leitungen sowie offene Eingänge direkt am Simulationsanfang als Fehler zu melden, sind solche Probleme recht leicht zu erkennen. Wird ein Leitungskurzschluss gemeldet, so ist dieser oft dadurch verursacht, dass ein Leitungstreiber auf die Busleitung zugeschaltet wird, bevor der letzte abgeschaltet worden ist.

Dieser Fehler kommt durch unterschiedliche Signallaufwege zustande und kann meist durch Verzögerungsglieder beseitigt werden.

Sollen nur reine logische Verknüpfungen getestet werden, so lässt sich die Funktion des skriptgesteuerten Testens anwenden. Dabei wird in einem Code festgelegt, wie sich bestimmte Werte bei bestimmten Eingaben zu verhalten haben. Beim Ausführen dieses Skripts testet die Software dann alle möglichen Kombinationen durch und liefert detaillierte Ergebnisse. Die Einbeziehung des Schaltungstakts ist hierbei ebenfalls möglich. Allerdings wird das Testen von komplexeren Konstrukten schon erheblich schwieriger. Daher wird diese Methode nur für einfache Teilschaltungen eingesetzt.

Eine Begutachtung durch Außenstehende hat dabei geholfen die Schaltungen hinsichtlich Verständnis und optischer Aufbereitung zu verbessern.

10 Rückblick auf die Projektplanung

10.1 Erfüllungsgrad der Anforderungsspezifikation

Die folgende Tabelle bewertet den Erfüllungsgrad der Festforderungen und optionalen Aufgaben aus der Anforderungsspezifikation.

Tabelle 5 – Bewertung des Erfüllungsgrades der Forderungen der Anforderungsspezifikation aus der Projektplanung

Forderung	Bewertung
Kernziele	
Entwicklung einer Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers.	Forderung ist vollständig erfüllt.
Anpassung der Entwicklung auf Lehrzwecke (pädagogische Verwendbarkeit, verständliche und nachvollziehbare Erklärungen).	Forderung ist weitgehend erfüllt. Die Hardware-Interruptverarbeitung wurde an die Lehrvorgehensweise angepasst, und die Visualisierung auf gutes Verständnis ausgerichtet. Zusätzliche Erklärungen und visuelle Effekte fördern das Verständnis der Studierenden.

Forderung	Bewertung
Die Visualisierung soll digital medial mithilfe einer Software durchgeführt werden.	Forderung ist vollständig erfüllt. Der Simulator Digital ist verwendet.
Die Interruptverarbeitung soll nach außen hin der des Intel 8086 Prozessors entsprechen.	Die Forderung ist erfüllt. Die Hardware-Interruptverarbeitung in der Simulation entspricht in vereinfachter Weise dem Vorgang des Intel 8086 Mikroprozessors.
Optionale Ziele	
Die Visualisierung sollte nach Möglichkeit in Einzelschritttaktung durchlaufen werden können.	Die Forderung ist erfüllt. Die Simulation lässt sich im Einzelschritt durchlaufen.
Falls durchführbar auch Darstellung der Prozesse innerhalb des Prozessors.	Die Forderung ist teilweise erfüllt. Zwar sind die Prozesse innerhalb der CPU einsehbar, diese weichen jedoch sehr stark von den realen Prozessen ab.
Vorzugsweise Verwendung des Intel 8086 Prozessors.	Diese Forderung ist nicht erfüllt. Der Prozessor in der Visualisierung ist nur ein Ersatzprozessor, der die Funktionalitäten für die Hardware-Interruptverarbeitung bereitstellt und nur im Kontext der Simulation betrieben werden kann.
Implementierung von Übungsaufgaben	Diese Forderung ist nicht erfüllt. Die Entwicklung von Übungsaufgaben hätte das Zeitmanagement überfordert.
So weit wie möglich als gekapselte Anwendung ausführbar.	Diese Forderung ist teilweise erfüllt. Zwar ist die Simulation ohne Fachwissen bedienbar, allerdings ist sie auch nicht schreibgeschützt und kann so bei unsachgemäßer Anwendung beschädigt werden.

Alle Kernziele des Projekts konnten erreicht werden. Lediglich einige optionale Ziele wurden nicht oder nur teilweise erreicht. Alle durch die Spezifikation geforderten Dokumente sind erstellt worden. Mögliche Risiken wurden weitgehend beseitigt. Die Testung verlief erfolgreich und zufriedenstellend. Die Bilanz ist somit positiv.

10.2 Vergleich der Zeitplanung mit dem tatsächlichen zeitlichen Projektablauf

In *Anhang 5 – Zeitvergleich* ist die Zeitplanung (hellblau) mit dem tatsächlichen zeitlichen Projektablauf (grün) verglichen. Zu sehen ist, dass einzelne Phasen deutlich gedehnt wurden. Insbesondere Phase IV hat wesentlich länger gebraucht als geplant und hat so die anderen Phasen nach hinten verschoben. Die Testung und der Projektabschluss haben quasi gleichzeitig stattgefunden. Letztendlich wurde der geplante Projektabschlusstermin jedoch nur um wenige Tage verfehlt.

11 Résumé

Gefolgt von einer umfangreichen Projektplanung hat die Recherche zur Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessors das Verständnis für den Verarbeitungsprozess herstellen können. Daraufhin folgten erste Testschaltungen mit der Software Digital. Diese ist als Simulationssoftware gegenüber einer Animationssoftware aufgrund der Nähe zur Technik bevorzugt worden. Es folgte die Entwicklung eines Prototyps und verschiedener Testschaltungen. Der Umgang mit dem Arbeitsspeicher wurde aus pädagogischen Gründen überdacht und geändert. In der Endversion der Visualisierung wurden alle wichtigen Informationen optisch hervorgehoben und ein klar strukturierter Einzelschrittablauf entworfen. Dabei wurden immer wieder einzelne Fehler aufgrund von Laufzeiterscheinungen behoben. Während der Entwicklung finden durchgehend Tests statt, die am Ende in der Testungsphase durch externe Gutachter verifiziert wurden. Zum Projektabschluss ist eine vollständig funktionsfähige und verständliche Visualisierung der Hardware-Interruptverarbeitung des Intel 8086 Mikroprozessor für Lehrzwecke entstanden. Die Visulisierung funktioniert auf Basis einer Simulation, die mit dem Simulator für digitale Schaltkreise „Digital“ entworfen ist.

Durch die Anwendung des Elementarprozessschema ETVX konnte die Qualität des Produkts durchgehend aufrechterhalten werden. Außerdem konnten alle geforderten Ziele erreicht werden. Beim Zeitmanagement besteht noch Optimierungsbedarf.

Trotz des soliden Endergebnisses gibt es noch viele Erweiterungs- und Verbesserungsmöglichkeiten. So wäre es beispielsweise möglich die Hardware-Interruptverarbeitung vollständig zu simulieren und nicht vor dem Start des

Interruptprogramms abzubrechen. Außerdem könnte man die Verarbeitung, soweit die Pädagogik es zulässt, auch noch deutlich detaillierter dargestellt werden. Um einen besseren Einblick in die inneren Prozesse der Interruptverarbeitung zu erhalten, kann der Prozessor noch universeller gestaltet werden. Der Einsatz einer echten Nachbildung des 8086 wäre hier zu überlegen. Durch die Entwicklung eines skriptgesteuerten Tests für die gesamte Simulation könnte weiterhin die Qualität effizienter abgesichert werden.

Das Durchlaufen der Simulation im Einzelschritt soll den Studierenden die Möglichkeit geben, die einzelnen Phasen länger zu beobachten. Gerade auch für die digitale Lehre hat dieses Projekt eine hohe Relevanz und kann so hoffentlich einen Vorteil für die Studierenden erzielen.

12 Literatur- und Quellenverzeichnis

[1] *Digital.* v0.26, 2021. [Online]. Verfügbar unter: https://github.com/hneemann/Digital

[2] P. Dr. Ing. Moos, „Vorlesung Informatik 1: Entwurfstechnik 2, Lebenszyklusmodelle“. WiSe 18/19. DHBW Mannheim, März 2019.

[3] M. Rafiquzzaman, *Fundamentals of digital logic and microcomputer design,* 5. Aufl. Hoboken, N.J.: Wiley, 2005.

[4] G. Schmitt, *Mikrocomputertechnik mit dem 16-Bit-Prozessor 8086: Maschinenorientierte Programmierung ; Grundlagen - Schaltungstechnik - Anwendungen,* 2. Aufl. München, Wien: Oldenbourg, 1989.

[5] L. B. Das, *The x86 microprocessors: 8086 to Pentium, multicores, atom, and the 8051 microcontroller*. New Delhi, India: Dorling Kindersley (India), 2014. [Online]. Verfügbar unter: http://proquest.tech.safaribooksonline.de/9789332540798

[6] T. Nguyen, *Procesoro Intel D8086 CS (Customer Sample).* [Online]. Verfügbar unter: https://commons.wikimedia.org/wiki/Category:Intel_8086?uselang=de (Zugriff am: 2. April 2021).

[7] K. FRICKE, *DIGITALTECHNIK: Lehr- und Übungsbuch für Elektrotechniker und Informatiker,* 8. Aufl. Wiesbaden: Springer Vieweg, 2018.

[8] Intel Corporation, „The 8086 Family User's Manual“, Santa Clara, CA 95051, Okt. 1979.

[9] G. Dipl. Ing. Lehmann, „Vorlesung Mikrocomputertechnik: Interruptsystem“. DHBW Mannheim, 2019.

[10] Intel Corporation, „8086 16-BIT HMOS MICROPROCESSOR 8086/8086-2/8086-1“, Sep. 1990.

[11] H. Neemann, „Digital“, 25. Jan. 2021. [Online]. Verfügbar unter: https://github.com/hneemann/Digital. Zugriff am: 6. Februar 2021.

Anhänge

Anhang 1

-

Projektkonzept

Projektkonzept

Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers zu Lehrzwecken mittels einer Software zur Simulation von digitalen Schaltungen

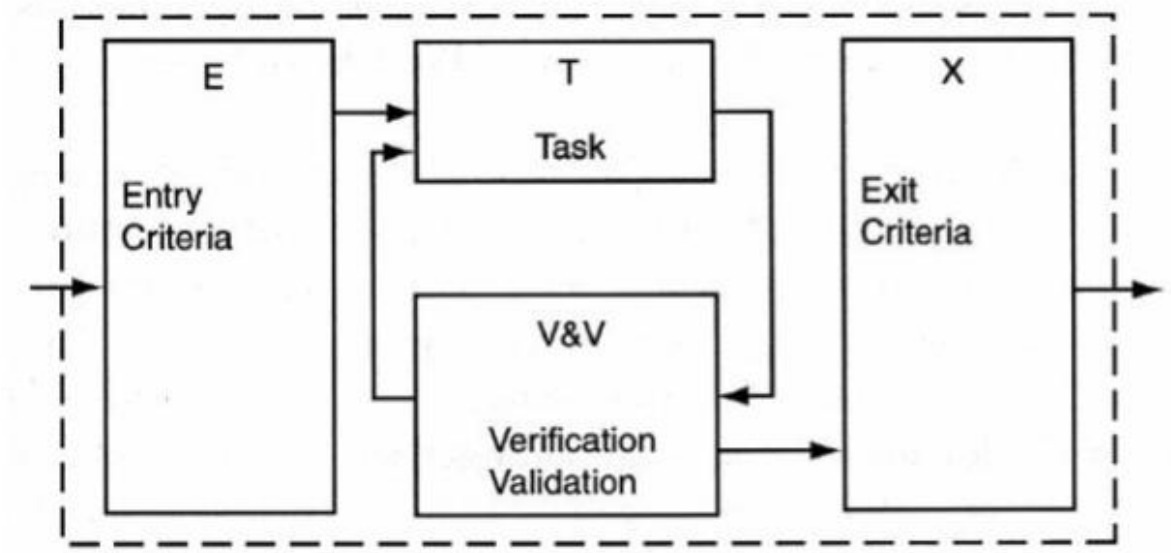
1. Anforderungsspezifikation

Siehe Dokument „Anforderungsspezifikation“.

2. Entwicklungsprozessschema

Grundsätzlich wird eine agile Entwicklung angestrebt. Die Kernzeile des Projekts bleiben im Fokus der Entwicklung und werden entsprechend der Rahmenbedingungen angepasst. Auf Herausforderungen wird situationsgerecht und interaktiv reagiert. Angestrebt sind schnelle und effiziente Lösungen im Sinne des Gesamtkonzepts.

Das Projekt soll anhand des Elementarprozessschemas ETVX bearbeitet werden.



Folgende Prozessphasen werden definiert.

Phase	I	II
Name	Projektvorbereitung	Entwurf
Eingangskriterium	Mit Beginn des Projekts	Abschluss von Phase I
Aufgabe	Anforderungsspezifikation und Zeitplan erstellen, Auswahl der Software, Festlegung des Projekttitels, Vorbereitung der Dokumente, Literaturrecherche	Ablauf der Interruptverarbeitung im Diagramm darstellen, Austesten der Fähigkeiten der Software inkl. Feststellung von Problembereichen, Ablauf der Entwicklung festlegen
Verifikation	Sind Anforderungen und Zeitplan realistisch umsetzbar? Ist die Software zum Erreichen der Kernziele sinnvoll? Spiegelt der Projekttitel die Ziele wieder? Mindestens 3 passende Quellen gefunden?	Entspricht die Darstellung der tatsächlichen Interruptverarbeitung und enthält sie alle wichtigen Informationen? Decken die erkannten Funktionen der Software alle Anforderungen der Interruptverarbeitung ab? Ist der Arbeitsaufwand realistisch und absehbar?
Ausgangskriterium	Die Aufgaben sind hinsichtlich der Validierungskriterien erfüllt. Es sind keine Frage mehr offen bezüglich der Aufgabenstellung und der Ziele.	Der Ablauf der Interruptverarbeitung in der Software ist vollständig festgelegt. Es besteht eine genaue Vorstellung über die Entwicklung der Simulation.

Phase	III	IV
Name	Entwicklung I	Entwicklung II
Eingangskriterium	Abschluss von Phase II	Abschluss von Phase III
Aufgabe	Entwicklung der einzelnen Komponenten die für die Interruptverarbeitung notwendig sind (Interruptcontroller, RAM, ROM, Verbindungen und Peripherie, CPU), Zwischentestung der einzelnen Komponenten, Planung des Zusammenbaus	Zusammenbau der Komponenten und Entwicklung der Schnittstellen, Optimierungen, um Einzelschritttaktung zu ermöglichen, Hinzufügen von erklärenden Elementen, Optionale Features
Verifikation	Funktionieren die Komponenten wie sie sollten? Ist die Umsetzung anschaulich und optisch ansprechend (ggf. eigene Graphik erstellen)? Sind Komplexität und schnelle Verständlichkeit ausgewogen? Gibt es Probleme beim Zusammenbau?	Funktioniert alles nach der Zusammensetzung noch wie zuvor? Fehlen Bauteile oder erfüllen Schnittstellen nicht ihren Zweck? Ist der Ablauf verständlich und erkennbar?
Ausgangskriterium	Alle Komponenten sind entwickelt und funktionstüchtig. Dem Zusammenbau steht nichts mehr im Wege.	Die Simulation bildet die Hardware-Interruptverarbeitung funktionsgetreu ab und ist verständlich. Die optische Entwicklung ist ebenfalls abgeschlossen. Mögliche Zusatzfunktionen sind ausgreift.

Phase	V	VI
Name	Testung	Projektabschluss
Eingangskriterium	Abschluss von Phase IV	Abschluss von Phase V
Aufgabe	Überprüfung der Simulation durch Tests hinsichtlich Funktionalität, Verständnis, Handhabung und Fehleranfälligkeit	Ausarbeitung von Anleitung und ggf. Übungsaufgaben, Organisatorische Aufarbeitung aller Projektinhalte, Überprüfung, ob die Projektziele unter Einhaltung der Vorgaben erreicht wurden, Feststellung von Entwicklungspotential und Nachlässigkeiten
Verifikation	Welche Fehler treten auf und wie können Sie behoben oder umgangen werden? Wird die Interruptverarbeitung den Studierenden durch die Visualisierung schneller verständlich?	Ist die Anleitung benutzerfreundlich? Ist alles erledigt? Was fehlt bzw. muss noch gemacht werden? Welche Verbessrungen wären angebracht? Was ist die größte Schwachstelle des Projekts?
Ausgangskriterium	Schwachstellen der Simulation sind erkannt und ggf. behoben. Im Regelfall funktioniert das Produkt wie es soll.	Das Projekt ist vollständig abgewickelt und es stehen keine Punkte mehr aus die zur Erfüllung der Aufgabenstellungen notwendig sind. Das Projekt wurde kritische reflektiert und Verbessrungsmöglichkeiten sind bekannt.

3. Zeitplanung

Das Projekt soll so schnell wie möglich, jedoch mit korrekter Ausführung aller Prozessschritte abgeschlossen werden. Dabei soll möglichst der erstelle Zeitplan eingehalten werden. Die konkrete Aufteilung der Zeiten sind im Dokument „Zeitplan“ festgehalten.

4. KPIs (Key Performance Indicators)

- Der Zeitplan wird nicht gedehnt (Die Bearbeitungszeit einer Phase ist kleiner als eine Woche)
- Die Schnittstelle zwischen Bit- und Hexadezimalsystem funktioniert
- Eingabe und Ausgabe funktionieren ordnungsgemäß
- Das Einbetten der Schaltungen verläuft erfolgreich
- Die Simulation ist ohne Fachwissen bedienbar
- Ein Interrupt lässt sich in der Visualisierung ohne Probleme begutachten und wird von externen verstanden

Anhang 2

-

Anforderungsspezifikation

Anforderungsspezifikation

Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers zu Lehrzwecken mittels einer Software zur Simulation von digitalen Schaltungen

1. Definition der Kernziele (Festforderungen)

- Entwicklung einer Visualisierung der Hardware-Interruptverarbeitung eines Mikrocomputers
- Anpassung der Entwicklung auf Lehrzwecke (pädagogische Verwendbarkeit, verständliche und nachvollziehbare Erklärungen)
- Die Visualisierung soll digital medial mithilfe einer Software durchgeführt werden
- Die Interruptverarbeitung soll nach außen hin der des Intel 8086 Prozessors entsprechen

2. Optionale Ziele (Wünsche)

- Die Visualisierung sollte nach Möglichkeit in Einzelschritttaktung durchlaufen werden können
- Falls durchführbar auch Darstellung der Prozesse innerhalb des Prozessors
- Vorzugsweise Verwendung des Intel 8086 Prozessors
- Implementierung von Übungsaufgaben
- Soweit wie möglich als gekapselte Anwendung ausführbar

3. Dateien und Dokumente

Folgende Dateien bzw. Dokumente sind zum Abschluss des Projektes in folgendem Umfang notwendig.

Name	Beschreibung	Dateiformat
Visualisierung der Hardware-Interruptverarbeitung	Fertige Visualisierung der Hardware-Interruptverarbeitung eines Intel 8086 Prozessors nach den Spezifikationskriterien	Anwendung bzw. Simulation innerhalb einer Anwendung
Anleitung zur Verwendung der Visualisierung	Beschreibung der Funktionsweise und Handhabung der entwickelten Anwendung/Simulation	PDF

4. Risiken

Da es unwahrscheinlich ist eine Software zu finden, welche alle Anforderungen exakt abbildet und der Zeitaufwand zur Programmierung einer komplett neuen Anwendung zu groß wäre, ist zu erwarten, dass vereinzelt Forderungen nicht komplett erfüllt werden können bzw. teilweise etwas unelegante Lösungen herangezogen werden müssen.

5. Testung

Die Testung der Visualisierung soll in einzelnen Komponenten sowie als zusammengefügtes Gesamtkonzept erfolgen. Dabei wird hinsichtlich verschiedener Kriterien wie Funktionalität, Realitätsbezug, Verständlichkeit und Handhabung getestet.

Anhang 3

-

Zeitplan


Today
25 Jan '21
01 Feb '21
08 Feb '21
15 Feb '21
22 Feb '21
01 Mar '21
08 Mar '21
15 Mar '21
22 Mar '21
Start
Mon 18.01.21
✔Vorlauf
Mon 18.01.21 - Thu 28.01.21
Phase I - Projektvorbereitung
Thu 21.01.21 - Sun 07.02.21
Phase II - Entwurf
Mon 08.02.21 - Sun 14.02.21
Phase III - Entwicklung I
Mon 15.02.21 - Thu 25.02.21
Phase IV - Entwicklung II
Fri 26.02.21 - Thu 04.03.21
Phase V - Testung
Fri 05.03.21 - Sun 14.03.21
Phase VI - Projektabschluss
Mon 15.03.21 - Fri 26.03.21
Finish
Fri 26.03.21

Anhang 4

-

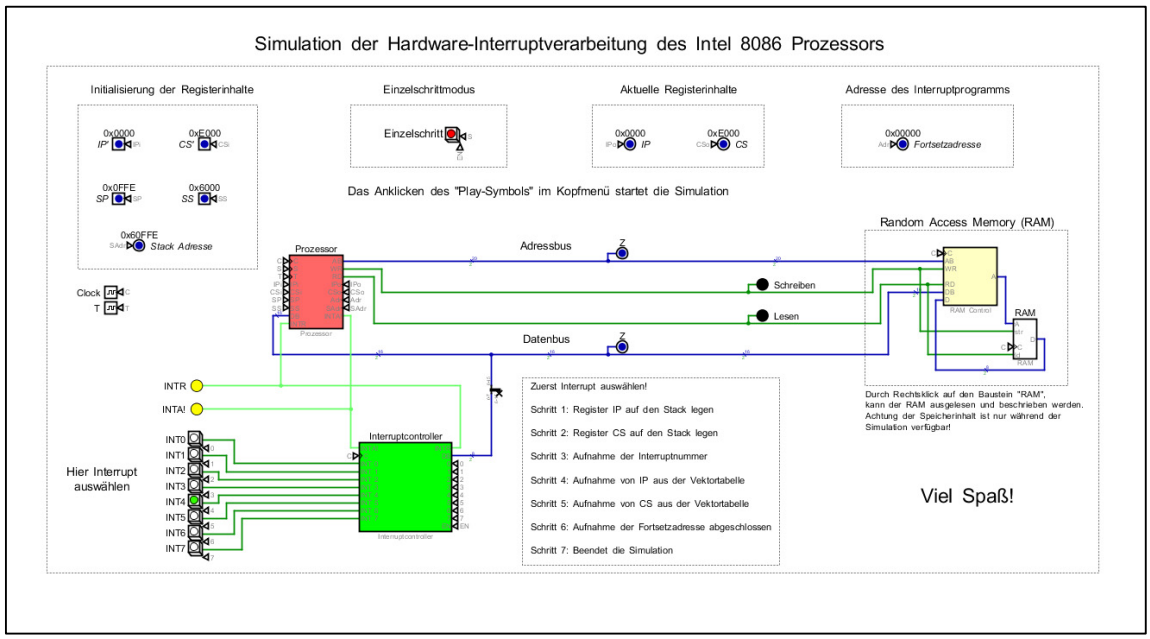
Bedienungsanleitung für die Simulation der Hardware Interruptverarbeitung des Intel 8086 Prozessors

Bedienungsanleitung

für die Simulation der Hardware-Interruptverarbeitung des Intel 8086 Prozessors

1. Einführung

Die vorliegende Simulation visualisiert die Hardware-Interruptverarbeitung des Intel 8086 Prozessors und basiert auf einer digitalen Schaltungssimulation. Als Simulationssoftware dient das Programm „Digital“, ein Simulator für digitale Schaltkreise.



2. Das Simulationsprogramm Digital

Die Verwendete Software mit dem Namen „Digital“ ist ein Simulator für digitale Schaltkreise und kostenfrei (GPL – General Public License) erhältlich.

Der Simulator kann unter folgendem Link von der Entwicklerplattform GitHub heruntergeladen werden.

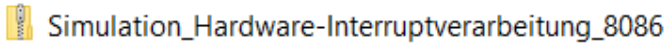
https://github.com/hneemann/Digital

Die Anwendung ist auf den Betriebssystemen Linux, Windows und MacOS lauffähig und kann ohne Installation verwendet werden.

Weitere Informationen zur Software können der Website unter obenstehendem Link entnommen werden.

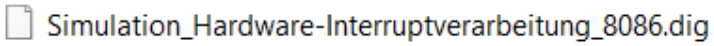
3. Öffnen der Simulationsdatei

Um die Simulation verwenden zu können, muss zuerst die Simulationsdatei in Digital geöffnet werden. Dazu ist es zuerst notwendig den komprimierten Ordner, welcher die Simulationsdateien enthält zu entpacken.

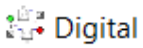


In diesem Ordner befindet sich in entpacktem Zustand ein weiterer Ordner mit dem Namen „Embedded_Circuits“. Dieser Ordner enthält einige Schaltungen, welche in die Hauptschaltung der Simulation eingebettet sind. Damit die Simulation der Hardware-Interruptverarbeitung problemlos verwendet werden kann, ist es zwingend notwendig, dass alle eigebetteten Schaltungen in diesem Ordner verbleiben und nicht entfernt oder verschoben werden.

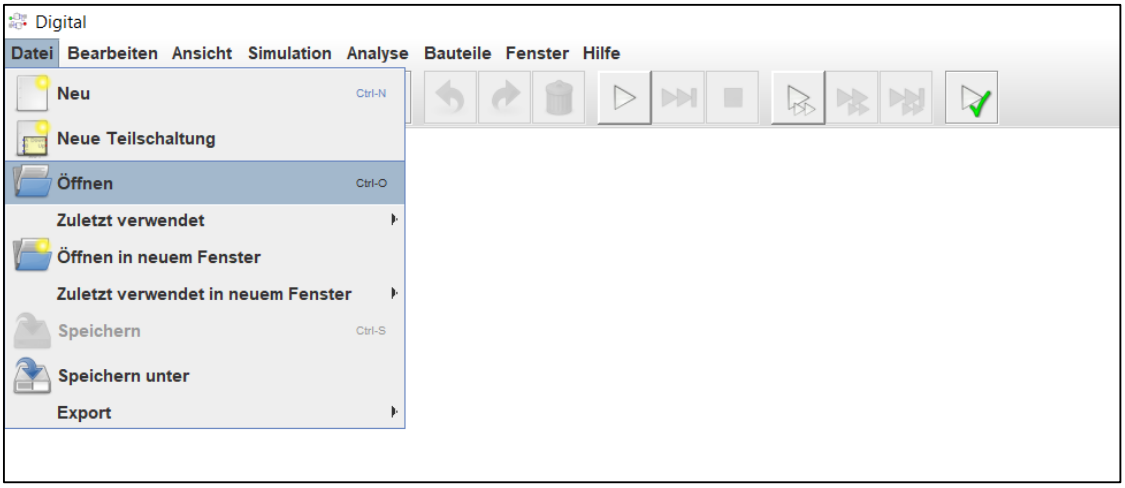
Zusätzlich ist neben dem Ordner mit den eingebetteten Schaltungen eine Datei zu finden. Diese Datei enthält die Hauptschaltung der Simulation und muss nun vom Programm aus geöffnet werden.

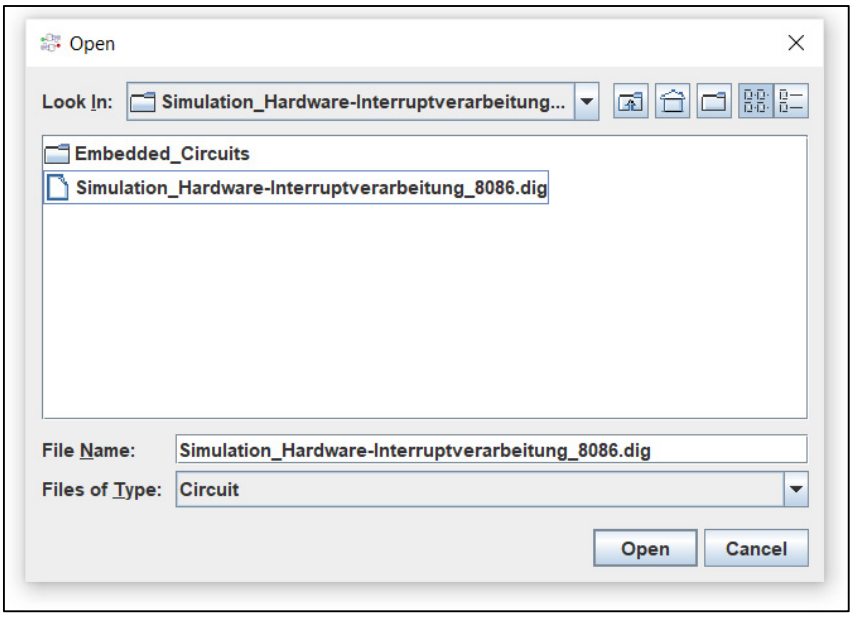


Zum Öffnen der Simulation ist zuerst das Programm Digital zu starten. Der Ordner, welcher von der GitHub Website heruntergeladen wurde, enthält zum Start der Applikation verschiedene Anwendungsdateien. Abhängig vom Betriebssystem muss hier nun die richtige Anwendung ausgeführt werden. Daraufhin öffnet sich das Programm direkt ohne Installation.



Zum Öffnen der Simulationsdatei wird in der oberen Menüleiste ganz links „Datei“ ausgewählt. In dem sich öffnenden Drop-down Menü kann nun über den Punkt „Öffnen“ die bereits erwähnte Hauptschaltung mit dem Namen „Simulation_Hardware-Interruptverarbeitung_8086.dig“ im Explorer ausgewählt werden.





Durch Klicken auf „Öffnen“ wird dann die Simulation geöffnet.

4. Anwendung der Simulation

Allgemeines:

Die Simulation und das zugehörige Modell sind nicht schreibgeschützt, da es in diesem Programm keine Differenzierung zwischen Anwender- und Entwicklungsumgebung gibt. Es ist somit möglich, Veränderungen jeglicher Art durchzuführen und somit folglich auch für die Funktionalität der Simulation schädliche. Es ist daher ratsam, vorsichtig zu agieren und/oder vorher eine Kopie der Originaldateien zu erstellen.

Zur Verwendung der Simulation ist es notwendig, das Programm Digital selbst in den grundlegenden Funktionen bedienen zu können. Die wichtigsten Bedieneigenschaften sind im Folgenden beschrieben.

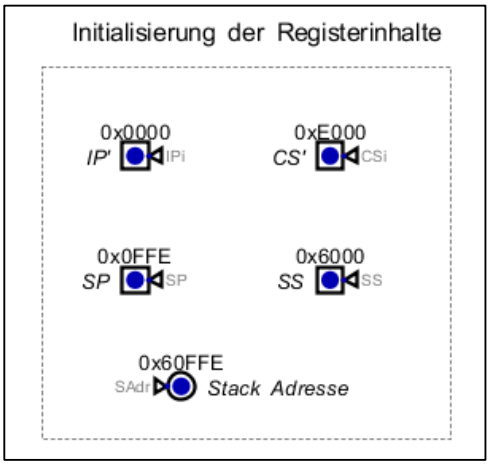
- Mit einem gehaltenen Rechtsklick auf den Schaltungshintergrund ist es möglich, das eigene Sichtfeld auf dem Schaltungshintergrund zu bewegen. So kann der Simulationsbereich in den Fokus gerückt werden.
- Durch Scrollen am Mausrad kann rein- oder rausgezoomt werden.
- Im Simulationsmodus können durch einen Klick auf Objekte der Schaltung verschiedene Aktionen durchgeführt werden.
- **Achtung!** Diverse Aktionen haben im Bearbeitungsmodus andere Auswirkungen wie im Simulationsmodus. Es sollte daher immer darauf geachtet werden, in welchem Modus man sich gerade befindet.

Vorbereiten der Simulation:

Um in den Simulationsmodus zu wechseln wird, auf das „Play-Symbol“ in der Kopfleiste des Programms geklickt. Durch den Knopf mit dem „Stopp-Symbol“ auf der rechten Seite kann die Simulation zu jedem Zeitpunkt gestoppt werden und das Programm kehrt zurück in den Bearbeitungsmodus.

Sobald der Simulationsmodus gestartet wurde, ist es möglich, eine Hardware-Interruptverarbeitung im Einzelschritt durchzugehen. Zuvor können optional noch einige Voreinstellungen vorgenommen werden.

1. Es ist möglich, die initialisierten Inhalte der Register IP, CS, SP und SS zu verändern. Dies ist im Feld „Initialisierung der Registerinhalte“ links oben in der Simulation möglich. Natürlich können auch einfach die Voreingestellten Inhalte beibehalten werden. Die Voreinstellungen sind in der folgenden Abbildung zu sehen.

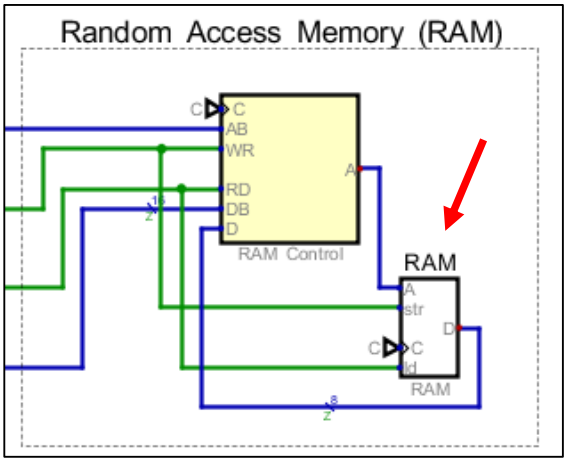


Außerdem wird unterhalb des Stapelsegments (SS – Stack Segment) und des Stapelzeigers (SP – Stack Pointer) direkt die sich daraus ergebende Stapeladresse (Stack Adresse) angezeigt.

Um die Inhalte der Register zu verändern ist, lediglich ein Klick auf die Eingangssymbole notwendig. Daraufhin erscheint ein Fenster, in dem der Wert angepasst werden kann.

2. Weiterhin können die Speicherzellen des RAM voreingestellt werden. Dies ist notwendig, um die Interruptvektortabelle zu initialisieren. Der RAM hat entsprechend des Adressbusses eine Größe von 1 MiB (1 Mega Binary Byte) und wird beim Wechseln in den Simulationsmodus programmseitig mit 00h je Speicherzelle beschrieben.

   Um den RAM händisch zu beschreiben, klickt man mit der Maus im Feld „Random Access Memory (RAM)“ auf den unteren rechten Baustein, der die Überschrift RAM trägt. Daraufhin öffnet sich ein neues Fenster mit einer Matrix, in der man dann jede beliebige Speicherzelle beschreiben kann.

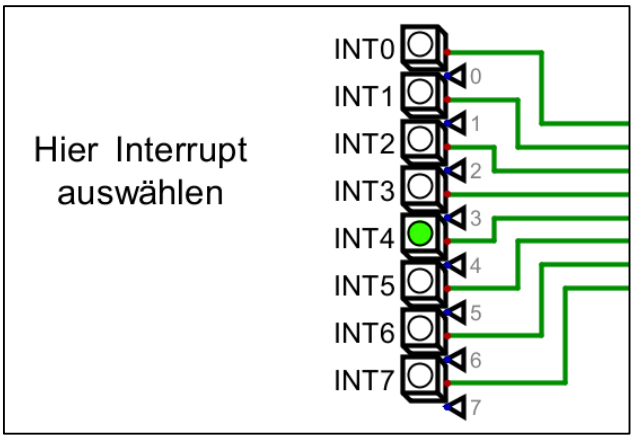


Ablauf der Simulation:

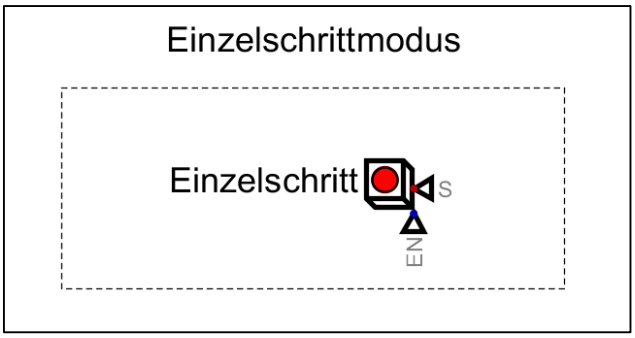
Die Simulation der Hardware-Interruptverarbeitung erfolgt im Einzelschrittmodus, sodass alle Verarbeitungsschritte in Ruhe nachvollzogen werden können.

Zuerst muss ein Hardware-Interrupt durchgeführt werden. Dazu stehen unten Links in der Hauptschaltung 8 Taster mit den Bezeichnungen INT0 bis INT7 zur Verfügung. Durch Drücken eines Tasters wird der jeweilige Interrupt ausgelöst und vom Interruptcontroller registriert. Der Taster des Interrupts leuchtet danach grün.

Achtung: Werden mehrere Taster nacheinander gedrückt, so ist immer der Interrupt aktiv der zuletzt ausgelöst wurde. Die Taster, die davor gedrückt wurden, haben keine Auswirkung mehr auf die Interruptverarbeitung.



Sobald ein Interrupt ausgelöst wurde, kann die Verarbeitung im Einzelschritt durchlaufen werden. Um einen Schritt weiterzugehen, wird der Taster „Einzelschritt“ im Feld „Einzelschrittmodus“ am oberen Rand der Schaltung verwendet. Dieser Taster leuchtet rot, sobald ein Interrupt ausgelöst wurde und indiziert dadurch, dass die Simulation nun bereit ist im Einzelschritt durchlaufen zu werden.



Durch jede Betätigung des Tasters geht die Simulation einen Schritt weiter in der Hardware-Interruptverarbeitung.

Folgende Schritte werden nacheinander ausgeführt.

Schritt 1 Register IP auf den Stack legen

In diesem Schritt schreibt der Prozessor den Inhalt des Registers IP mit seinem initialisierten Wert auf den Stapel im RAM.

Schritt 2 Register CS auf den Stack legen

In diesem Schritt schreibt der Prozessor den Inhalt des Registers CS mit seinem initialisierten Wert auf den Stapel im RAM.

Schritt 3 Aufnahme der Interruptnummer

In diesem Schritt nimmt der Prozessor die vom Interruptcontroller auf den Datenbus gelegte Interruptnummer entsprechend des ausgelösten Interrupts auf.

Schritt 4 Aufnahme von IP aus der Vektortabelle

In diesem Schritt liest der Prozessor entsprechend der aufgenommenen Interruptnummer den neuen Wert für das Register IP aus der Interruptvektortabelle im RAM.

Schritt 5 Aufnahme von CS aus der Vektortabelle

In diesem Schritt liest der Prozessor entsprechend der aufgenommenen Interruptnummer den neuen Wert für das Register CS aus der Interruptvektortabelle im RAM.

Schritt 6 Aufnahme der Fortsetzadresse abgeschlossen

Nach Schritt 5 ist durch die Aufnahme der neuen Werte für IP und CS die Adresse für das Interruptprogramm bekannt. Die Aufnahme der Fortsetzadresse ist somit abgeschlossen und der Prozessor geht in diesem Schritt zurück in den Ruhezustand über. Die weitere Hardware-Interruptverarbeitung ist ab diesem Punkt nicht weiter simuliert, da nun ein zuvor geschriebenes Interruptprogramm abgearbeitet würde. Auch das Zurückholen der ursprünglichen Inhalte des IP und CS Registers vom Stapel

wird nicht mehr simuliert. Der Simulierte Teil der Hardware-Interruptverarbeitung ist daher an dieser Stelle beendet.

Schritt 7 Beendet die Simulation

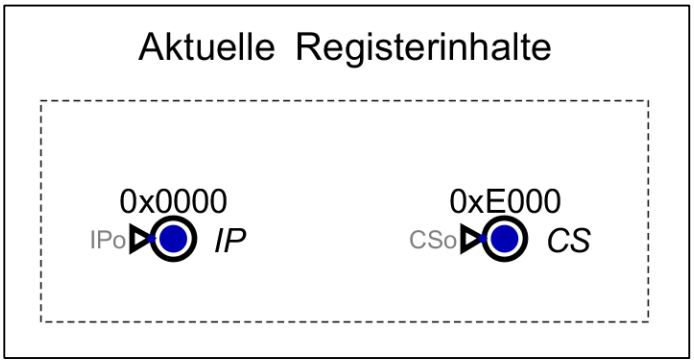
Ein siebtes Betätigen des Einzelschritttasters beendet die Simulation vollständig. Das Programm kehrt dabei aus dem Simulationsmodus zurück in den Bearbeitungsmodus.

Diese Einzelschritte sind in ihrer Kurzform auch in der Schaltung wieder zu finden.

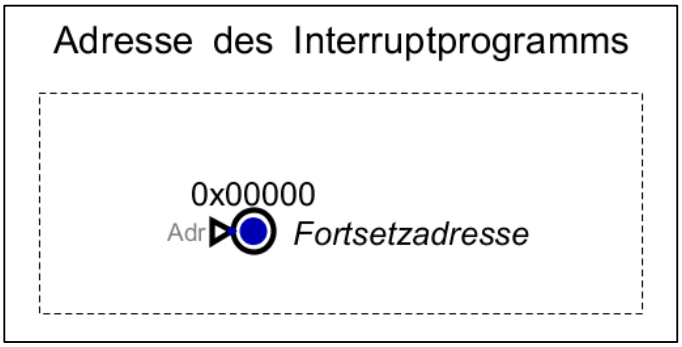
Optische Indikatoren:

Um eine bessere Visualisierung zu erreichen, enthält die Schaltung bzw. die Simulation einige optische Indikatoren.

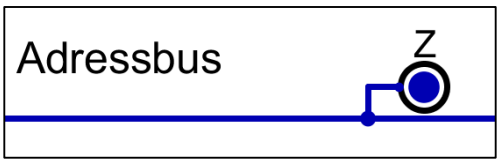
1. Im Feld „Aktuelle Registerinhalte“ können die Inhalte der Register IP und CS zu jedem Zeitpunkt während der Simulation beobachtet werden.



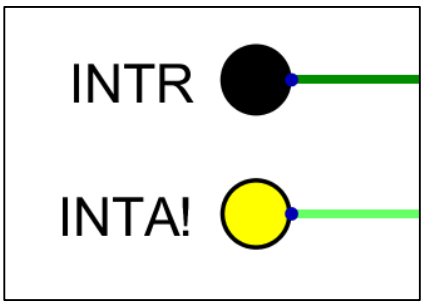
2. Im Feld „Adresse des Interruptprogramms“ kann am Ende der Simulation die Fortsetzadresse für das Interruptprogramm, ermittelt aus den Registern CS und IP, abgelesen werden.



3. Grundsätzlich gilt für Leitungen in Digital während der Simulation Folgendes.
    - Eine 1 Bit Leitung ist hellgrün, wenn ihr Zustand „high“ ist. Sie erscheint dunkelgrün, wenn ihr Zustand „low“ ist.
    - Ein Bus wird ebenfalls als eine Leitung dargestellt, jedoch wird an dieser die Anzahl der Adern angezeigt. Der Zustand bzw. Wert eines Buses während der Simulation kann durch grüne Zeichen an diesem festgestellt werden.
4. Der Zustand des Adressbusses und des Datenbusses kann zusätzlich durch jeweils eine Anzeige beobachtet werden. Der Wert des Busses wird dabei hexadezimal angegeben, wobei der Buchstabe „Z“ für hochohmig steht.



5. Die Zustände der Leitungen für das Lesen und Schreiben von oder in den RAM sowie die Zustände der Leitungen INTR (Interrupt) und INTA! (Interrupt Acknowledge) werden zusätzlich durch Leuchtkreise angezeigt. Ist der Kreis farbig so ist die Leitung „high“, ist er schwarz so ist die Leitung „low“.



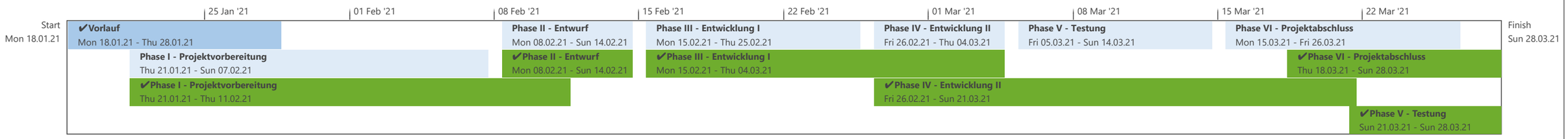
Viel Spaß beim Simulieren der Hardware-Interruptverarbeitung des Intel 8086 Prozessors

Silas Gerhard

Anhang 5

-

Zeitvergleich


25 Jan '21
01 Feb '21
08 Feb '21
15 Feb '21
22 Feb '21
01 Mar '21
08 Mar '21
15 Mar '21
22 Mar '21
Start
Mon 18.01.21
Finish
Sun 28.03.21
✔Vorlauf
Mon 18.01.21 - Thu 28.01.21
Phase I - Projektvorbereitung
Thu 21.01.21 - Sun 07.02.21
✔Phase I - Projektvorbereitung
Thu 21.01.21 - Thu 11.02.21
Phase II - Entwurf
Mon 08.02.21 - Sun 14.02.21
✔Phase II - Entwurf
Mon 08.02.21 - Sun 14.02.21
Phase III - Entwicklung I
Mon 15.02.21 - Thu 25.02.21
✔Phase III - Entwicklung I
Mon 15.02.21 - Thu 04.03.21
Phase IV - Entwicklung II
Fri 26.02.21 - Thu 04.03.21
✔Phase IV - Entwicklung II
Fri 26.02.21 - Sun 21.03.21
Phase V - Testung
Fri 05.03.21 - Sun 14.03.21
✔Phase V - Testung
Sun 21.03.21 - Sun 28.03.21
Phase VI - Projektabschluss
Mon 15.03.21 - Fri 26.03.21
✔Phase VI - Projektabschluss
Thu 18.03.21 - Sun 28.03.21