* Nennen Sie kurz die Phasen der Übersetzung!
* Beschreiben Sie die Phasen der Übersetzung ausführlicher.
* Wie werden Tokens genau erkannt?
* Beschreiben Sie ausführlicher, wie Gleitkommazahlen, natürliche Zahlen und Strings erkannt werden.
* Warum kann der Parser nicht mit regulären Sprachen wie a^n b^n arbeiten? Thema pumping Lemma.
* Thema Top-Down-Analyse und Problematik mit Backtracking.
* Linksfaktorisierung.
* Was sind LL(k)-Grammatiken?
* Kann man statt LL(1) auch LL(2) nutzen?
* First- und Follow-Menge.
* Kontextsensitive Grammatiken. Wozu nutzt man sie? Semantische Analyse: Typüberprüfung.
* Unterschied zwischen synthetisierten und vererbten Attributen.
* Wann geschieht die semantische Analyse: simultan oder nach der syntaktischen Analyse?
* Wann genau wird das Attribut A(X) gesetzt?
* Optimierungsmethoden.
* Die Phasen der Übersetzung aufzählen und zu jeder Phase kurz etwas erklären
  + Lexikalische Analyse:
    - Tokendefinition, endlicher Automat, was sind reguläre Sprachen
  + Syntaktische Analyse:
    - warum reicht hier der endliche Automat nicht mehr aus (habe ich anhand der Klammern erklärt, die syntaktisch immer in der gleichen Anzahl vorhanden sein müssen. Das ist dann wie eine Sprache a^n b^n, die nicht regulär ist)
* Wann ist eine Grammatik eindeutig/mehrdeutig?
  + Bekanntes Beispiel für mehrdeutige Grammatik: dangling else Problem (kannte ich nicht)
  + - Top-Down Analyse erklären, wie heißen die Grammatiken?
  + Bottom-Up Analyse erklären und Grammatik Bezeichnung
* Semantische Analyse
  + Was passiert hier? (Stichwort Typüberprüfung, Auflösung von Überladenen Operatoren)
* Zwischencodeerzeugung
  + Hier habe ich etwas zu dem Zweck dieser Phase erzählt und welche Arten von Zwischencode es gibt und woran sie sich unterscheiden (abstrakter Syntaxbaum, DAG, 3AC, Postfix Notation)
* Codeoptimierung
  + Welche Arten gibt es? (Lokal, global, schleifen)
  + Optimierungsmethoden aufzählen und kurz erklären, wie sie funktionieren
* Welche Phasen hat die Übersetzung?
  + Wie funktioniert das?
    - Analysephase:
      * Lexikalische
        + Was ist ein regulärer Ausdruck?
        + Was ist eine reguläre Sprache? Meinerseits verweise auf die induktive Definitionen.
      * Syntaktische
        + Welche Arten gibt es? Top Down Bottom Up. Bisschen die Unterschiede erklärt.
        + Kann jede Programmiersprache auf Basis von kontextfreien Grammatiken übersetzt werden?( Das wusste ich nicht so genau. Hab gesagt nein und das war auch richtig. Siehe semantische Phase!)
      * Semantische
        + Was ist attributierte Grammatik?

Hier hing er dann weiter an der Frage zu welcher Sprachklasse die denn gehören? Das einzige was mir einfiel was höher war als kontextfrei waren von Turing Maschinen erkennbare. War aber falsch: Kontextsensitive Sprache(Also die von Turing Maschinen erkennbaren mit beschränkten Band glaub ich)

Wofür brauchen wir die Attribute?

Eine Attributgrammatik ist eine kontextfreie Grammatik, die um Attribute sowie Regeln und Bedingungen erweitert ist. Angewandt wird das Konzept im Compilerbau, um beispielsweise die Einhaltung von Regeln zu überprüfen, die mit kontextfreien Grammatiken nicht formuliert werden können. Solche Regeln sind z. B. die, dass jede Variable deklariert sein muss und ihrem Datentyp entsprechend verwendet wird“ Im Nachhinein hat er auch angemerkt, dass der Teil der Semantikanalyse wohl mein Schwachpunkt war.

* + - Synthesephase
    - Zwischencodeerzeugung
      * Welchen ZC kennen Sie? Hab 3AC genannt und kurz was dazu gesagt
    - Optimierung Hier wollte er jeweils n Beispiel für eine Optimierung
      * Algebraische: Hab gesagt Zusammenfassung von arithmetischen Ausdrücken
      * Maschinenunabhängige: Z.B Löschen von nicht verwendeten Variablen!
      * Prozessorspezifisch: Fiel mir nichts ein, er schlug Registerallokation vor, also Anpassung an die Anzahl der Register des Prozessors
* Welche Übersetzungsphasen gibt es?
  + Welche Aufgabe hat die lexikalische Analyse? Zwischendurch habe ich die Handimplementierung eines Scanners auf Basis von Zustandsdiagrammen (NEAs) erläutert.
  + Welche Aufgabe hat die Syntaxanalyse?
    - Welche beiden Strategien gibt es?
    - Was setzt jede Variante der Top-down-Analyse voraus?
    - Wodurch wird vorausschauende sackgassenfreie Analyse möglich?
    - Welche Verfahren gibt es bei der Bottom-up-Analyse?
    - Warum parst man mit kontextfreien Sprachen?
    - Welche Parser sind mächtiger und warum?
    - Beschreiben Sie die LR-Analyse.
  + Warum gibt es neben der syntaktischen noch die semantische Analyse?
    - Was ist eine attributierte Grammatik?
  + Wozu braucht man Symboltabellen?
* Welche Codeoptimierungsverfahren gibt es?
* DEAs auch zu einem zusammenfassen könnte.
* Syntaktische Analyse.
  + Drauf los erzählt über Bottom-up und Top Down. Dann wurde ich direkt gefragt was Nach und Vorteile der beiden Varianten sind. Ich habe dann im Endeffekt ausführlich LL(1) erklärt. Bis ich mit der Erklärung der Berechnung FIRST und FOLLOW angefangen habe.
  + KF-Grammatiken
  + Wieso sind Programmiersprachen nicht regulär? Sind alle Programmiersprachen kontextfrei,- (nein kontextsensitiv, Behelf mit attributierter Grammatik
* „Die Übersetzung ist unterteilt in Analyse / Synthese. Nennen Sie doch mal die 3 Phasen der Analyse“? Lexikalische Analyse, syntaktische Analyse, semantische Analyse
* „Was macht man bei der lexikalischen Analyse genau“? Input String wird zu Tokens umgewandelt. Token mit Beispiele (Bezeichner, Keywords, Konstanten,..)
* „Es geht also darum Tokens zu erkennen – zeichnen und erklären Sie einen Automat der Integer erkennt.“
  + Automat (ist kein DEA übrigens) gezeichnet der Integer erkennt. Ich hab einen gezeichnet der noch 0 unterscheidet (keine führende 0) und halt erklärt.
* „Was macht man bei der syntaktischen Analyse genau?“ Syntax wird analysiert, also wie die Token zusammenhängen. Input: Tokenmenge Output: Syntaxbaum
* „Welche Verfahren gibt es zur Syntaxanalyse?“ Top-Down und Bottom-Up
* „Was zeichnet das Top Down Verfahren aus?“ Syntaxbaum wird von der Wurzel aus mit dem Startsymbol der Grammatik entwickelt.
  + „Wie funktioniert das konkret?“ Als Beispiel Recursive Descent Verfahren erklärt.
* „Welche Probleme können hier auftreten?“ Linksrekursion und dadurch Endlosschleife, Beispielproduktionsregel aufgeschrieben
* „Wie kann man das Problem der Linksrekursion lösen?“ Wie im Skript Linksrekursion auflösen am Beispiel schriftlich gezeigt.
* „Linksfaktorisierung – welches Problem löst das und zeigen Sie eine“ Gleiche Terminalsymbole bei verschiedenen Optionen -> keine Entscheidung durch Lookahead möglich. Wieder schriftliches Beispiel gezeigt und aufgelöst.
* „Was heißt eigentlich LL(k) und was ist das für eine Grammatik?“
  + Formale Definition (Steuermengen paarweise disjunkt) + Erklärung der beiden Ls (von links, Linksableitung) und k(Lookahead)
* „Was ist die schon angesprochene Steuermenge?“
  + Steuermenge ist Konkatenation von FIRST und FOLLOW, FIRST und FOLLOW an Beispielregeln erklärt ohne formale Definition
* „Wie berechnet man diese Mengen?“
  + Algorithmen zur Berechnung der Mengen erklären – ich hatte hier bei FOLLOW einen anderen Algorithmus als aus dem Skript erklärt, der wohl ineffizienter war, aber auch korrekt. Diese Frage ging relativ lang, sollte man also auf jeden Fall erklären können!
* „Bottom-Up Verfahren – was zeichnet dieses aus?“
  + Baum wird von den Blättern (Tokens) aus aufgebaut.
* „Bei Bottom Up benutzt man Shift-Reduce Parser, welche Aktionen gibt es und wie funktioniert das?“ Die vier Aktionen Shift, Reduce, Accept, Error erklärt – kleines Beispiel mit Shift und Stack aufgezeichnet.
* „Was versteht man unter einem Handle?“ Formale Definition aufgeschrieben und erklärt.
* „Erklären Sie kurz die Operatorvorranganalyse.“
  + Shift / Reduce Konflikt erklärt -> macht Entscheidung notwendig. Die drei Relationen kurz erklärt, Aufstellen der Tabelle, vor jeder Aktion wird die Relation auf dem Stack mit zu lesendem Token verglichen und jeweilige Aktion durchgeführt.
* „Was sind die Anforderungen dafür, dass die Operatorvorranganalyse möglich ist?“ Es gibt 3 Kriterien (ich wusste nur eps):
  + 1. Keine Epsilon Produktion,
  + 2. Keine aufeinanderfolgenden Terminale rechte Seite,
  + 3. Keine gleichen rechten Seiten.
* „Welche 3 Phasen hat die Synthese?“
  + Zwischencodegenerierung
  + Codeoptimierung
  + Codegenerierung
* „Was kann man sich unter Zwischencode vorstellen?“
  + Beispiele: AST, DAG, Postfix, 3AC 21. „3 Adress Code – können Sie kurz erklären was das ist?“ An Assembler angelehnter Code bei dem maximal 3 Adressen auftreten können.
* „Nennen Sie je ein Beispiel mit Erklärung für maschinenabhängige / -unabhängige Optimierung.“
  + Constant Folding / Propagation als unabhängige Optimierung mit Beispiel erklärt.
  + Peephole Optimisation mit Beispielen aus Skript (LOAD Zeile im Assembler sparen, überflüssige GOTO Anweisung) erklärt und aufgeschrieben
    - „Wieso heißt das eigentlich Peephole Optimization?“ Weil nur ein sehr kleiner zusammenhängender Ausschnitt des Maschinencodes betrachtet wird.
* Erklären Sie mal Backtracking
* Beseitigung von Linksrekursion
* Wie gehen Steuermengen, First, Follow
* Warum terminiert das
* Wie geht Bottom-up
* Was ist ein Handle
* Erklären Sie mal die Operator-Vorrang-Analyse
* Was gibt es an lokalen und/oder maschinenabhängigen Optimierungen?
* Linksfaktorisierung - was bedeutet LL(k)
* Was sind Steuer-, First-, und Follow-Mengen?
* Wie werden First- und Follow-Mengen berechnet?
* Wie funktioniert die Operator-Vorrang-Analyse?
* Was genau ist 3AC? - Ein paar 3AC-Befehle aufschreiben