1. ΓΕΝΙΚΑ

ΣΧΟΛΗ	**ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**		
ΤΜΗΜΑ	**ΠΛΗΡΟΦΟΡΙΚΗΣ**		
ΕΠΙΠΕΔΟ ΣΠΟΥΔΩΝ	**ΠΡΟΠΤΥΧΙΑΚΟ**		
ΚΩΔΙΚΟΣ ΜΑΘΗΜΑΤΟΣ	**306ΕΥΥΚ**	**ΕΞΑΜΗΝΟ ΣΠΟΥΔΩΝ**	**3ο**
ΤΙΤΛΟΣ ΜΑΘΗΜΑΤΟΣ	**ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ**		

ΑΥΤΟΤΕΛΕΙΣ ΔΙΔΑΚΤΙΚΕΣ ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ *σε περίπτωση που οι πιστωτικές μονάδες απονέμονται σε διακριτά μέρη του μαθήματος π.χ. Διαλέξεις, Εργαστηριακές Ασκήσεις κ.λπ. Αν οι πιστωτικές μονάδες απονέμονται ενιαία για το σύνολο του μαθήματος αναγράψτε τις εβδομαδιαίες ώρες διδασκαλίας και το σύνολο των πιστωτικών μονάδων*	**ΕΒΔΟΜΑΔΙΑΙΕΣ ΩΡΕΣ ΔΙΔΑΣΚΑΛΙΑΣ**	**ΠΙΣΤΩΤΙΚΕΣ ΜΟΝΑΔΕΣ**
Διαλέξεις	**2**	**5**
Φροντιστηριακές Ασκήσεις	**1**	
Προσθέστε σειρές αν χρειαστεί. Η οργάνωση διδασκαλίας και οι διδακτικές μέθοδοι που χρησιμοποιούνται περιγράφονται αναλυτικά στο 4.		

ΤΥΠΟΣ ΜΑΘΗΜΑΤΟΣ *Υποβάθρου , Γενικών Γνώσεων, Επιστημονικής Περιοχής, Ανάπτυξης Δεξιοτήτων*	Επιστημονικής Περιοχής
ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ ΜΑΘΗΜΑΤΑ:	-
ΓΛΩΣΣΑ ΔΙΔΑΣΚΑΛΙΑΣ και ΕΞΕΤΑΣΕΩΝ:	Ελληνική
ΤΟ ΜΑΘΗΜΑ ΠΡΟΣΦΕΡΕΤΑΙ ΣΕ ΦΟΙΤΗΤΕΣ ERASMUS	-
ΗΛΕΚΤΡΟΝΙΚΗ ΣΕΛΙΔΑ ΜΑΘΗΜΑΤΟΣ (URL)	-

2. ΜΑΘΗΣΙΑΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Μαθησιακά Αποτελέσματα

Περιγράφονται τα μαθησιακά αποτελέσματα του μαθήματος οι συγκεκριμένες γνώσεις, δεξιότητες και ικανότητες καταλλήλου επιπέδου που θα αποκτήσουν οι φοιτητές μετά την επιτυχή ολοκλήρωση του μαθήματος.

Συμβουλευτείτε το Παράρτημα Α

- *Περιγραφή του Επιπέδου των Μαθησιακών Αποτελεσμάτων για κάθε ένα κύκλο σπουδών σύμφωνα με Πλαίσιο Προσόντων του Ευρωπαϊκού Χώρου Ανώτατης Εκπαίδευσης*
- *Περιγραφικοί Δείκτες Επιπέδων 6, 7 & 8 του Ευρωπαϊκού Πλαισίου Προσόντων Διά Βίου Μάθησης*

και Παράρτημα Β

- *Περιληπτικός Οδηγός συγγραφής Μαθησιακών Αποτελεσμάτων*

Σκοπός του μαθήματος είναι να παρουσιάσει βασικά θέματα της τεχνολογίας των Compilers στον Μηχανικό Πληροφορικής.

Με την επιτυχή ολοκλήρωση του μαθήματος ο φοιτητής/τρια θα είναι σε θέση:

- Να κατανοεί την έννοια των Μεταγλωττιστών και των βασικών προσεγγίσεων στην υλοποίηση γλωσσών προγραμματισμού και συγκεκριμένα τόσο της αρχιτεκτονικής των παραδοσιακών μεταγλωττιστών (π.χ. GNU gcc, g++ compilers), όσο και τα πλαίσια που βασίζονται σε εικονικές μηχανές (virtual machines) π.χ. Java, LLVM).
- Να κατανοεί το scanning και τα lexers, και τις τυπικές προσεγγίσεις στην κατασκευή lexers που βασίζονται σε πεπερασμένα αυτόματα.
- Να κατανοεί την έννοια του parsing, τις τυπικές προσεγγίσεις που βασίζονται σε γραμματικές, LL Grammars, LR Grammars.
- Να αναπτύσσει βασικές μορφές ενδιάμεσης αναπαράστασης, π.χ. abstract syntax trees, directed acyclic graphs, control flow graphs, Static single assignment form, Stack Machines.

- Να εξοικειωθεί ως case studies με την αναπαράσταση της Java Virtual Machine και της Low Level Virtual Machine.
- Να αναπτύσσει τα βασικά χαρακτηριστικά της assembly language με χρήση ως case studies την X86/x64 assembly και την ARM64 assembly.
- Να αναπτύσσει τις βασικές προσεγγίσεις και τους αλγορίθμους για code generation και optimization.

Γενικές Ικανότητες

Λαμβάνοντας υπόψη τις γενικές ικανότητες που πρέπει να έχει αποκτήσει ο πτυχιούχος (όπως αυτές αναγράφονται στο Παράρτημα Διπλώματος και παρατίθενται ακολούθως) σε ποια / ποιες από αυτές αποσκοπεί το μάθημα;.

Αναζήτηση, ανάλυση και σύνθεση δεδομένων και πληροφοριών, με τη χρήση και των απαραίτητων τεχνολογιών	*Σχεδιασμός και διαχείριση έργων*
Προσαρμογή σε νέες καταστάσεις	*Σεβασμός στη διαφορετικότητα και στην πολυπολιτισμικότητα*
Λήψη αποφάσεων	*Σεβασμός στο φυσικό περιβάλλον*
Αυτόνομη εργασία	*Επίδειξη κοινωνικής, επαγγελματικής και ηθικής υπευθυνότητας και ευαισθησίας σε θέματα φύλου*
Ομαδική εργασία	*Άσκηση κριτικής και αυτοκριτικής*
Εργασία σε διεθνές περιβάλλον	*Προαγωγή της ελεύθερης, δημιουργικής και επαγωγικής σκέψης*
Εργασία σε διεπιστημονικό περιβάλλον	
Παράγωγή νέων ερευνητικών ιδεών	

- Αυτόνομη Εργασία
- Ομαδική Εργασία

3. ΠΕΡΙΕΧΟΜΕΝΟ ΜΑΘΗΜΑΤΟΣ

1. Εισαγωγή στους Μεταγλωττιστές (Compilers)

Τονίζεται η χρησιμότητα του μαθήματος για την καλύτερη κατανόηση πλήθος βασικών εννοιών που έχουν να κάνουν με την σχεδίαση και λειτουργία των γλωσσών προγραμματισμού. Ταυτόχρονα ο φοιτητής μελετά και σημαντικές τεχνικές και προγραμματιστικά εργαλεία με ευρύτερη εφαρμογή σε πολλές εφαρμογές, εκτός φυσικά των μεταγλωττιστών.

Παρουσιάζεται η δομημένη προσέγγιση στην κατασκευή των μεταγλωττιστών με διάσπαση του πολύπλοκου έργου της υλοποίησης ενός μεταγλωττιστή σε πολλαπλές φάσεις ανάλυσης.

2. Λεκτική Ανάλυση (Lexical Analysis)

Πρόκειται για το έργο του μετασχηματισμού της ανεπεξέργαστης σειράς χαρακτήρων ενός προγράμματος σε μια ροή από tokens (token stream). Η κατασκευή των Λεκτικών Αναλυτών χρησιμοποιεί βασικές έννοιες της Επιστήμης των Υπολογιστών με σημαντική χρησιμότητα σε πλήθος εφαρμογών.

Συγκεκριμένα εισάγεται η έννοια των *Κανονικών Εκφράσεων (Regular Expressions)* και των συσχετιζόμενων Μαθηματικών Μοντέλων των *Μη-αιτιοκρατικών Πεπερασμένων Αυτομάτων (Nondeterministic Finite Automata)* και των *Αιτιοκρατικών Πεπερασμένων Αυτομάτων (Deterministic Finite Automata)*. Παρουσιάζεται η συστηματική κατασκευή λεκτικών αναλυτών βασισμένων στα παραπάνω μαθηματικά εργαλεία, με την πιθανή χρησιμοποίηση και αυτόματων εργαλείων λογισμικού (lexer generators).

3. Συντακτική Ανάλυση (Syntax Analysis)

Η φάση της συντακτικής ανάλυσης πραγματοποιεί το σημαντικότατο έργο της κατασκευής του *δέντρου σύνταξης (syntax tree)* από την ροή tokens που λαμβάνεται από τον λεκτικό αναλυτή. Η εύρεση των συντακτικών σφαλμάτων (syntax errors) του προγράμματος είναι αρμοδιότητα της παρούσας φάσης.

Η έννοια των *γραμματικών (grammars)* αποτελεί την θεωρητική βάση στην κατασκευή συντακτικών αναλυτών. Η κατηγορία των *γραμματικών ανεξάρτητων συμφραζόμενων (context free grammars)* είναι προεξέχουσας σημασίας. Αναλύονται μέθοδοι κατασκευής δέντρων σύνταξης (syntax trees) με *σαρωτές (parsers)* βασιζόμενους σε τυπικές γραμματικές. Ο χειρισμός της *ασάφειας (ambiguity)* και η υλοποίηση της επιθυμητής *προτεραιότητας τελεστών* (*operator precedence*) αποτελούν σημαντικά θέματα. μελετώνται τα συναφή θέματα του *LL(1) Parsing, LR Parsing, Recursive Descent Parsing* και *SLR Parsing*.

4. Εμβέλειες και Πίνακες Συμβόλων (Scopes and Symbol Tables)

Οι σύγχρονες γλώσσες προγραμματισμού, ιδιαίτερα οι αντικειμενοστραφείς, παρέχουν την δυνατότητα καθορισμού της εμβέλειας στην οποία κάθε μεταβλητή είναι ορατή. Ο μεταγλωτιστής χρησιμοποιώντας ευέλικτες δομές *πινάκων συμβόλων* (symbol tables) υλοποιεί τους αντίστοιχους κανόνες ορατότητας (ή εμβέλειας) μεταβλητών της γλώσσας.

5. Διερμήνευση (Interpretation)
Στο σημείο αυτό, της κατασκευής δηλαδή του δέντρου σύνταξης και των συσχετισμένων πινάκων συμβόλων, γίνεται δυνατή η *διερμήνευση* (*interpretation*) του προγράμματος, δίχως την παραγωγή εκτελέσιμου κώδικα. Το έργο αυτό πραγματοποιείται με μια συστηματική επίσκεψη στους κόμβους του δέντρου σύνταξης (*syntax tree traversal)*. Με την διερμήνευση του κώδικα παρέχεται η δυνατότητα ευέλικτης αποσφαλμάτωσης (debugging). Παρά ταύτα, η ταχύτητα της διερμηνευμένου κώδικα δεν επιδέχεται σύγκριση με μεταφρασμένο κώδικα. Για τον λόγο αυτό, οι interpreters περιορίζονται μόνο στην φάση ανάπτυξης και πρωτοτυποποίησης συστημάτων και για την "συγκόληση" (gluing) μεταφρασμένων τμημάτων κώδικα.

6. Έλεγχος Τύπων (Type checking)
Στο σημείο αυτό εισάγεται η έννοια της *σημασιολογικής ανάλυσης (semantic analysis),* που έχει ως στόχο την ανίχνευση σφαλμάτων που έχουν να κάνουν με την σημασιολογία των εκφράσεων και όχι απλά της σύνταξης. Ένα εργαλείο για ανίχνευση μερικών τέτοιων σφαλμάτων είναι ο συσχετισμός *τύπων* με μεταβλητές και η επιβολή κανόνων ορθής χρησιμοποίησης και σύνθεσης τύπων.
Η κατασκευή μιας γλώσσας ως *statically typed ,* προαπαιτεί από τον μεταγλωττιστή τον υπολογισμό τύπων για όλες τις μεταβλητές. Παραδοσιακά αυτό γίνεται με *δηλώσεις τύπων (type declarations)*, αν και πολλές σύγχρονες γλώσσες συνδυάζουν και συστήματα για αυτόματη ανακάλυψη του τύπου (*type inference systems*).
Οι *dynamically typed* γλώσσες αναβάλουν τον καθορισμό των τύπων των μεταβλητών μέχρι τον χρόνο εκτέλεσης (run time). Η τεχνολογία τους είναι πολύ διαφορετική. Αναπτύσσονται οι βασικές προσεγγίσεις υλοποίησης του "duck typing" και αλγόριθμοι για σχετικά αποτελεσματική εκτέλεση δυναμικού κώδικα (dynamically typed κώδικα). Παραδοσιακά χρησιμοποιούνται σχήματα caching, αλλά σύγχρονα περιβάλλοντα εκτέλεσης, όπως η σημερινή Java (Java 12), παρέχουν και βελτιστοποιημένη υποστήριξη σε χαμηλό επίπεδο (π.χ. εντολή invoke dynamic της Java 7+).
Άλλη καινοτόμα προσέγγιση, είναι το Just-In-Time compilation μετά από την αναγνώριση των τύπων σε χρόνο εκτέλεσης. Τέτοια προσέγγιση χρησιμοποιεί η σύγχρονη γλώσσα επιστημονικού προγραμματισμού Julia, που πετυχαίνει ταχύτητα που προσεγγίζει τις statically typed γλώσσες (π.χ. C/C++), παρόλο που είναι dynamically typed, κάνοντας just-in-time compilation στο πλαίσιο της τεχνολογίας LLVM (Low Level Virtual Machine).

7. Ενδιάμεση Παραγωγή Κώδικα (Intermediate Code Generation)
Η παραγωγή μιας ευέλικτης μορφής κώδικα από το δέντρο σύνταξης διευκολύνει εξαιρετικά το δύσκολο έργο της *βελτιστοποίησης* (*optimization)* και της παραγωγής του τελικού κώδικα (*code generation).* Μελετώνται παραδείγματα μετάφρασης προγραμματιστικών δομών σε ενδιάμεσες μορφές κώδικα. Στο ίδιο πλαίσιο είναι και η αντιστοίχιση σύνθετων δομών δεδομένων (structures, classes), πινάκων (arrays) και δηλώσεων συναρτήσεων/μεθόδων.

8. Παραγωγή Κώδικα (Code Generation)
Η παραγωγή του τελικού κώδικα από την ενδιάμεση μορφή, περιπλέκει πολλά νέα προβλήματα, τα οποία έχουν σχέση και με την αρχιτεκτονική του hardware για την οποία προορίζεται ο κώδικας. Μελετώνται θέματα όπως η *κατανομή καταχωρητών (register allocation),* η κατασκευή των activation records για την κλήση μεθόδων, η ανάλυση liveness μεταβλητών και γίνεται εισαγωγή σε σχήματα για
data-flow ανάλυση και βελτιστοποίηση του κώδικα.

4. ΔΙΔΑΚΤΙΚΕΣ και ΜΑΘΗΣΙΑΚΕΣ ΜΕΘΟΔΟΙ - ΑΞΙΟΛΟΓΗΣΗ

ΤΡΟΠΟΣ ΠΑΡΑΔΟΣΗΣ *Πρόσωπο με πρόσωπο, Εξ αποστάσεως εκπαίδευση κ.λπ.*	Στην τάξη
ΧΡΗΣΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ *Χρήση Τ.Π.Ε. στη Διδασκαλία, στην*	Εξειδικευμένο open source Λογισμικό σε περιβάλλον Linux

Εργαστηριακή Εκπαίδευση, στην Επικοινωνία με τους φοιτητές	
ΟΡΓΑΝΩΣΗ ΔΙΔΑΣΚΑΛΙΑΣ *Περιγράφονται αναλυτικά ο τρόπος και μέθοδοι διδασκαλίας.* *Διαλέξεις, Σεμινάρια, Εργαστηριακή Άσκηση, Άσκηση Πεδίου, Μελέτη & ανάλυση βιβλιογραφίας, Φροντιστήριο, Πρακτική (Τοποθέτηση), Κλινική Άσκηση, Καλλιτεχνικό Εργαστήριο, Διαδραστική διδασκαλία, Εκπαιδευτικές επισκέψεις, Εκπόνηση μελέτης (project), Συγγραφή εργασίας / εργασιών, Καλλιτεχνική δημιουργία, κ.λπ.* *Αναγράφονται οι ώρες μελέτης του φοιτητή για κάθε μαθησιακή δραστηριότητα καθώς και οι ώρες μη καθοδηγούμενης μελέτης ώστε ο συνολικός φόρτος εργασίας σε επίπεδο εξαμήνου να αντιστοιχεί στα standards του ECTS*	***Δραστηριότητα*** — ***Φόρτος Εργασίας Εξαμήνου*** Διαλέξεις — 26 x 2 = 52 ώρες Φροντιστηριακές Ασκήσεις που εστιάζουν στην εφαρμογή — 13 x 2 = 26 ώρες Αυτοτελής Μελέτη — 45 ώρες Γραπτές Εξετάσεις — 2 x 1 = 2 ώρες ***Σύνολο Μαθήματος (25 ώρες φόρτου εργασίας ανά πιστωτική μονάδα)*** — ***125 ώρες***
ΑΞΙΟΛΟΓΗΣΗ ΦΟΙΤΗΤΩΝ *Περιγραφή της διαδικασίας αξιολόγησης* *Γλώσσα Αξιολόγησης, Μέθοδοι αξιολόγησης, Διαμορφωτική ή Συμπερασματική, Δοκιμασία Πολλαπλής Επιλογής, Ερωτήσεις Σύντομης Απάντησης, Ερωτήσεις Ανάπτυξης Δοκιμίων, Επίλυση Προβλημάτων, Γραπτή Εργασία, Έκθεση / Αναφορά, Προφορική Εξέταση, Δημόσια Παρουσίαση, Εργαστηριακή Εργασία, Κλινική Εξέταση Ασθενούς, Καλλιτεχνική Ερμηνεία, Άλλη / Άλλες* *Αναφέρονται ρητά προσδιορισμένα κριτήρια αξιολόγησης και εάν και που είναι προσβάσιμα από τους φοιτητές.*	Γραπτή τελική εξέταση (100%)

5. ΣΥΝΙΣΤΩΜΕΝΗ-ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Alfred Aho, Monica S. Lam, Ravi Sethi, Jefrey D. Ullman, Compilers, Principles, Techniques, & Tools, Second Edition, Addison-Wesley, 2007
2. Michael L. Scott, Programming Language Pragmatics, Elsevier, 2006
3. Torben Egidius Mogensen, Introduction to Compiler Design, Springer-Verlag, 2011