KARTA OPISU PRZEDMIOTU - SYLABUS

Nazwa przedmiotu
Programowanie niskopoziomowe [N1Inf1>PLOW]

Przedmiot

Kierunek studiów
Informatyka

Rok/Semestr
2/3

Studia w zakresie (specjalność)
–

Profil studiów
ogólnoakademicki

Poziom studiów
pierwszego stopnia

Język oferowanego przedmiotu
polski

Forma studiów
niestacjonarne

Wymagalność
obligatoryjny

Liczba godzin

Wykład
12

Laboratorium
12

Inne
0

Ćwiczenia
0

Projekty/seminaria
0

Liczba punktów ECTS

2,00

Koordynatorzy

dr inż. Wojciech Complak
wojciech.complak@put.poznan.pl

Wykładowcy

Wymagania wstępne

Student rozpoczynający ten przedmiot powinien posiadać podstawową wiedzę z algorytmiki i programowania w językach imperatywnych. Powinien posiadać umiejętność rozwiązywania podstawowych problemów z zakresu projektowania, sprawdzenia poprawności i implementowania algorytmów w języku C oraz umiejętność pozyskiwania informacji ze wskazanych źródeł. Powinien również rozumieć konieczność poszerzania swoich kompetencji / mieć gotowość do podjęcia współpracy w ramach zespołu. Ponadto w zakresie kompetencji społecznych student musi prezentować takie postawy jak uczciwość, odpowiedzialność, wytrwałość, ciekawość poznawcza, kreatywność, kultura osobista, szacunek dla innych ludzi.

Cel przedmiotu

1. Przekazanie studentom podstawowej wiedzy z zakresu programowania niskopoziomowego w oparciu o język C i asembler. Przedstawienie funkcji wewnętrznych kompilatora, architektury x86-64 z punktu widzenia programisty aplikacyjnego, podstaw asemblera w zakresie wstawek do programów w języku C oraz elementów binarnego interfejsu aplikacji. 2. Rozwijanie umiejętności posługiwania się zintegrowanym środowiskiem programistycznym w zakresie wsparcia dla programowania niskopoziomowego. 3. Przedstawienie specyficznych mechanizmów sprzętowych i sposobów ich wykorzystania za pomocą funkcji wewnętrznych i asemblera.

Przedmiotowe efekty uczenia się

Wiedza:
1. ma uporządkowaną i podbudowaną teoretycznie wiedzę ogólną w zakresie kluczowych zagadnień informatyki, oraz wiedzę szczegółową w zakresie architektury systemów komputerowych i imperatywnych języków programowania
2. ma podstawową wiedzę o cyklu życia systemów informatycznych, zarówno sprzętowych jak i programowych, a w szczególności o analizie wykonalności systemu oraz projektowaniu szczegółowym
3. zna podstawowe techniki, metody oraz narzędzia wykorzystywane w procesie rozwiązywania zadań informatycznych w zakresie analizy algorytmów, architektury systemów komputerowych i implementacji algorytmów

Umiejętności:
1. potrafi - zgodnie z zadaną specyfikacją - zaprojektować, sformułować specyfikację funkcjonalną w formie przypadków użycia na poziomie formalnego opisu, sformułować wymagania pozafunkcjonalne dla charakterystyki czasowej oraz zaimplementować oprogramowanie z wykorzystaniem imperatywnego języka programowania z pomocą odpowiednich technik i narzędzi
2. ma umiejętność formułowania algorytmów i ich implementacji z użyciem języka imperatywnego i zintegrowanego środowiska programistycznego

Kompetencje społeczne:
1. rozumie, że w informatyce wiedza i umiejętności bardzo szybko stają się przestarzałe
2. ma świadomość znaczenia wiedzy w rozwiązywaniu problemów inżynierskich oraz zna przykłady i rozumie przyczyny wadliwie działających systemów informatycznych, które doprowadziły do poważnych strat finansowych, społecznych lub też do poważnej utraty zdrowia, a nawet życia

Metody weryfikacji efektów uczenia się i kryteria oceny

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:

Efekty uczenia się przedstawione wyżej weryfikowane są w następujący sposób:
Ocena formująca:
a) w zakresie wykładów:
- na podstawie odpowiedzi na pytania dotyczące materiału omówionego na poprzednich wykładach
b) w zakresie laboratoriów:
- na podstawie oceny bieżącego postępu realizacji zadań
Ocena podsumowująca:
Sprawdzanie założonych efektów kształcenia realizowane jest przez:
- ocenę umiejętności związanych z realizacją ćwiczeń laboratoryjnych,
- ocenianie ciągłe, na każdych zajęciach (sprawdziany wejściowe) - premiowanie przyrostu umiejętności posługiwania się poznanymi zasadami i metodami,
- ocenę wiedzy i umiejętności związanych z realizacją zadań projektowych/laboratoryjnych poprzez kolokwia podczas trzeciego (język C) i piątego (asembler, asembler + język C) laboratorium; zadania mają zarówno charakter konstrukcyjny (np. napisz program) jak i analityczny (np. jaka będzie odpowiedź danego programu).
Uzyskiwanie punktów dodatkowych za aktywność podczas zajęć, a szczególnie za:
- omówienia dodatkowych aspektów zagadnienia,
- efektywność zastosowania zdobytej wiedzy podczas rozwiązywania zadanego problemu,
- uwagi związane z udoskonaleniem materiałów dydaktycznych,
- wskazywanie trudności percepcyjnych studentów umożliwiające bieżące doskonalenia procesu dydaktycznego.

Treści programowe

Program wykładu obejmuje następujące zagadnienia:
- język C, funkcje wewnętrzne, programowanie dla systemu Windows
- architektura i asembler X86-32, wplatania kodu asemblerowego do języka C
- obliczenia zmiennoprzecinkowe z wykorzystaniem FPU, obliczenia równoległe z wykorzystaniem MMX i SSE

Program laboratorium obejmuje następujące zagadnienia:
- programowanie w języku C: przenośne i dla systemu Windows, wykorzystanie funkcji wewnętrznych, środowisko MS Visual Studio
- programowanie hybrydowe: język C i wstawki w asemblerze X86-32
- programowanie hybrydowe: obliczenia zmiennoprzecinkowe i równoległe

Tematyka zajęć

Pierwszy wykład poświęcony jest omówieniu organizacji zajęć oraz omówieniu/przypomnieniu podstawowych konstrukcji języka C. Na zajęciach laboratoryjnych przygotowywane jest środowisko, omawiane zasady obsługi, tworzenia i uruchamiania programów w języku C.
Drugi wykład poświęcony jest bardziej zaawansowanym mechanizmom języka C (wersja C11) takim jak struktury, unie, pola bitowe oraz reprezentacja maszynowa. Prezentowane są również funkcje wewnętrzne kompilatora zarówno wieloplatformowe, jak i specyficzne dla architektury X86. Na zajęciach laboratoryjnych studenci tworzą złożone programy w języku C analizując reprezentację danych i możliwości funkcji wewnętrznych.
Trzeci wykład poświęcony jest prezentacji architektury X86-64, podstawom asemblera, instrukcjom arytmetycznym, logicznym i organizacji przepływu sterowania oraz zasadom wplatania kodu asemblerowego do języka C. Na zajęciach laboratoryjnych studenci zaczynają pisać proste wstawki asemblerowe.
Czwarty wykład dotyczy mechanizmu stosu, konwencji wywołań podprogramów, przekazywania parametrów i zasad tworzenia podprogramów w języku asemblera. Na zajęciach laboratoryjnych studenci tworzą podprogramy w języku asemblera.
Piąty wykład to prezentacja zaawansowanych mechanizmów przetwarzania (FPU, rozkazy multimedialne i wektorowe) dostępnych na platformie x86-64 i posługiwania się nimi z wykorzystaniem funkcji wewnętrznych kompilatora języka C i instrukcji asemblera. Pokazywane są przykłady uproszczenia kodu i poprawy wydajności. Ostatni wykład poświęcony jest omówieniu wykorzystania binarnego interfejsu aplikacji do wywołania usług systemu oraz podsumowaniu przedmiotu.

Metody dydaktyczne

1. wykład: prezentacja multimedialna, prezentacja ilustrowana przykładami podawanymi na tablicy, rozwiązywanie zadań, demonstracja narzędzi programistycznych,
2. ćwiczenia laboratoryjne: rozwiązywanie zadań, dyskusja.

Literatura

Podstawowa
1. Język ANSI C, B. W. Kernighan, D. M. Ritchie, Wydawnictwa Naukowo-Techniczne, dowolne wydanie
2. Mikroprocesory 80286, 80386 i i486, R. Goczyński, M. Tuszyński, Komputerowa Oficyna Wydawnicza HELP, 1991
3. Koprocesory arytmetyczne 80287 i 80387 oraz jednostka arytmetyki zmiennoprzecinkowej mikroprocesora i486, M. Tuszyński, R. Goczyński. Komputerowa Oficyna Wydawnicza HELP, 1992
4. MSDN, https://msdn.microsoft.com/
Uzupełniająca
1. Język C. Szkoła programowania, Wydanie VI, Prata S., Helion, 2016
2. Asembler w koprocesorze, S. Kruk, Wydawnictwo MIKOM, 2003
3. Mikroprocesor 8086, Z. Mroziński, Wydawnictwa Naukowo-Techniczne, 1992
4. Koprocesor arytmetyczny 8087, Z. Mroziński, Wydawnictwa Naukowo-Techniczne, 1992
5. Język C w pigułce. Kompletny przewodnik, Peter Prinz, Tony Crawford, Helion, 2017
6. The Art of 64-Bit Assembly, Volume 1, Randall Hyde, No Starch Press, 2021
7. Modern X86 Assembly Language Programming, 3rd Edition, Daniel Kusswurm, Apress, 2023

Bilans nakładu pracy przeciętnego studenta

	Godzin	ECTS
Łączny nakład pracy	50	2,00
Zajęcia wymagające bezpośredniego kontaktu z nauczycielem	24	1,00
Praca własna studenta (studia literaturowe, przygotowanie do zajęć laboratoryjnych/ćwiczeń, przygotowanie do kolokwiów/egzaminu, wykonanie projektu)	26	1,00