Wydziałowa Komisja Akredytacji Przedmiotów (WKAP)

Autor*:* dr hab. inż. Mariusz Rawski, mgr inż. Aleksander Pruszkowski

Systemy komputerowe: architektura i programowanie

Poziom kształcenia: I stopień
Forma i tryb prowadzenia przedmiotu: stacjonarna
Kierunek studiów: Cyberbespieczeństwo (CB)
Specjalność:
Grupa przedmiotów:
Poziom przedmiotu: podstawowy
Status przedmiotu: obowiązkowy
Język przedmiotu: polski
Semestr nominalny (tylko dla przedmiotów obowiązkowych): 4
Minimalny numer semestru: 4
Wymagania wstępne, zalecane przedmioty poprzedzające: SYCY, PRMT2
Limit liczby studentów: 60

Powód zgłoszenia przedmiotu: program studiów na nowym kierunku Cyberbezpieczeństwo

Cel przedmiotu:
Głównym celem przedmiotu jest kompleksowe zapoznanie studentów(ek) z budową systemów komputerowych oraz z aspektami programistycznymi związanymi z warstwami oprogramowania stykającymi się bezpośrednio ze sprzętem.

Treść kształcenia: łącznie ok 700 znaków na 1 ECTS nie mniej niż 1500 znaków

WYKŁADY:
1. Język C/C++ w systemie procesorowym
1.1. Budowa programów, rola funkcji w języku C/C++ w tym funkcji main()
1.2. Proste typy danych, deklaracje zmiennych określonego typu, zasięg zmiennych, zaawansowane typy danych: struktury, enumcje, ...
1.3. Rola preprocesora, prototypy funkcji, podział kodu na moduły

2. **Procesor w systemie komputerowym**
2.1. Przestrzenie pamięciowe (z punkt widzenia procesora, dla architektur: Harward, Von Neumna)
2.2. Rozkazy procesora (podział na: RISC, CISC, kodowanie rozkazów, tryby adresowania, porównanie listy rozkazów: X86 i NIOS II)

3. Pamięci
3.1. Typy, hierarchie i budowa pamięci używanych w systemach komputerowych (SRAM, DRAM/DDRxx, Flash/NOR/NAND, ...)
3.2. Mechanizmy zabezpieczenia przez kody korekcyjne zawartości pamięci komputerowych
3.3. Kodowanie i składowanie zmiennych typów podstawowych i zaawansowanych w pamięciach systemu komputerowego, rola stosu i sterty
3.4. Kodowanie zmiennych języków wysokiego poziomu (NKB, U2, IEEE754)
3.5. Konsolidacja kodu aplikacji, podstawowe formaty plików wykonywalnych

4. Cykl życia systemu komputerowego
4.1. Uruchamianie kodu i usuwanie błędów z kodu, narzędzia diagnostyczne (debuger, JTAG)
4.2. Procedury POST/BIOS, uruchamianie systemu komputerowego
4.3. Programy ładujące (Bootloader)
4.4. Proces uruchamiania kodu wynikowego przez procesor

5. System operacyjny dla systemów komputerowych
5.1. Definicja, budowa i działanie systemu Linux jako przykład nowoczesnego systemu operacyjnego
5.2. Styk aplikacji użytkowych z systemem operacyjnym, biblioteki jądra i systemowe (furtki systemowe, syscall)
5.3. Podstawy tworzenia oprogramowanie w języku C/C++ dla systemu Linux

6. Przerwania, obsługa sytuacji wyjątkowych
6.1. Rola przerwań w systemach komputerowych
6.2. Typowe sytuacje wyjątkowe w systemach komputerowych
6.3. Metody implementacji obsługi przerwań z przykładami w języku C/C++

7. Urządzenia wejścia / wyjścia w systemach komputerowym
7.1. Styk procesora z układami wejścia / wyjścia – warstwa sprzętu (łączenie, mapa pamięci)
7.2. Tworzenie oprogramowania współpracującego z peryferiami na przykładzie systemu operacyjnego Linux. Analiza dwóch podejść w tworzeniu kodu współpracującego ze sprzętem: podejście uproszczone - sterownik przestrzeni użytkownika i podejście zaawansowane - sterownik przestrzeni jądra

8.Sprzętowe wspomaganie systemu operacyjnego
8.1. Systemy zarządzania pamięciami (MMU), tryby chroniony, wirtualizacja pamięci
8.2. Realizacja fizyczna systemów wielowątkowych i wielo procesorowych

9. Zaawansowane układy peryferyjne
9.1. Układy wspierania działania pamięci: układy odświeżania pamięci, układy realizacji pamięci podręcznej, układy bezpośredniej komunikacji peryferii z pamięciami (DMA)
9.2. Pamięci masowe, klasyfikacja pamięci masowych, budowa dysków (HDD, SSD), mechanizmy zwiększania pojemności i niezawodności (JBOD, RAID, LVM)
9.3. Interfejsy komunikacji między elementami systemu komputerowego (PCI/PCIe, USB, karty sieciowe)
9.4. Karty graficzne, akceleratory graficzne (CUDA)
9.5. Akceleratory kryptograficzne, generatory liczb losowych, szyfrowanie strumieni danych
9.6. Koegzystencja procesorów i układów peryferyjnych w jednym układzie scalonym (SoC)

ĆWICZENIA:
-

LABORATORIA:
Zajęcia laboratoryjne mają być wprowadzeniem do wykonywanego projektu. Podzielono zajęcia te na 3 części tematyczne:
1) Zapoznanie się z projektowaniem systemów wykorzystujących procesor FPGA (np.: NIOS II): budowa projektu procesora w języku opisu sprzętu (np.: VHDL), utworzenie

minimalnego programu w C/C++ którego zadaniem ma być weryfikacja poprawności działania systemu przy założeniu braku systemu operacyjnego, uruchomienie całości z demonstracją jego działania.
2) Do powstałego w ramach 1 zajęć laboratoryjnych systemu, dodać układ peryferyjny zadany przez prowadzącego oraz napisać w C/C++ prosty demonstrator działania tego układu peryferyjnego.
3) Osadzić system Linux na wytworzonym w ramach 1 i 2 zajęć laboratoryjnych systemie oraz napisać odpowiedni sterownik przestrzeni użytkownika pozwalający na interakcję z zasobem peryferyjnym powstałym w ramach 2 zajęć laboratoryjnych.

PROJEKT:
Zbudować system procesorowy bazujący na podejściu bez systemu operacyjnego (tzw. bare-metal). Dodać do tego systemu wskazane przez prowadzących specjalistyczne peryferia (np.: układ wsparcia kryptografii, układ przyspieszenia operacji graficznych, transformaty FFT, obrazów i podobne, ...). Dla celów testowych wymagane jest utworzenie aplikacji testującej demonstrującej możliwości tak przygotowanego systemu.

ZAJĘCIA ZINTEGROWANE:
-

Treść kształcenia - streszczenie w jęz. angielskim:

The main objective of the course is to introduce students to the basics of computer architecture. Software aspects connected with hardware will also be discussed.

Egzamin: NIE

Literatura i oprogramowanie:
Materiały do zajęć – slajdy, opracowania, artykuły

Literatura
1. B. W. Kernighan, D. M. Ritchie, „Język ANSI C. Programowanie”, Helion, 2010, ISBN 978-83-246-2578-9
2. J. Ogrodzki, „Wstęp do systemów komputerowych”, OWPW 2005, ISBN: 83-7207-529-8

Literatura uzupełniająca
1. J. Corbet, A. Rubini, G. Kroah-Hartman, „Linux device drivers”, O'Reilly, 2005, ISBN 978-0-5965-5538-2
2. R. Pełka, „Mikrokontrolery, architektura, programowanie, zastosowania”, WKŁ, Warszawa 2000
3. W. Daca, „Mikrokontrolery – od układów 8-bitowych do 32-bitowych”, MIKOM, Warszawa 2000
4. M. Gołaszewski, „Procesory Nios II w układach FPGA”, Elektronika Praktycznie: 10/2010, 11/2010, 12/2010, 1/2011, 3/2011
5. Pong P. Chu, „Embedded SoPC Design with Nios II Processor and Verilog Examples”, John Wiley & Sons 2012, ISBN-13: 9781118011034

6. M. Nowakowski, „PicoBlaze. Mikroprocesor w FPGA”, BTC 2009, ISBN: 978-83-60233-55-9

Oprogramowanie

1. Pakiet dla tworzenia opisu FPGA oraz testowania tworzonych systemów: „Intel Quartus II Web Edition Software”
2. Pakiet dla tworzenia opisu procesorów NIOS II: „Nios II Embedded Design Suite”
3. Zestaw kompilator i linker dla tworzenia oprogramowania w językach C/C++ „GCC, the GNU Compiler Collection”
4. Zestaw do tworzenia podstawowego systemu plików z kolekcjami narzędzi dla systemu Linux „Buildroot”

Wymiar godzinowy zajęć:	W	C	L	P
	30	-	15	15

Przewidywane formy kształcenia i organizacja przedmiotu

Zajęcia będące wykładami będą wprowadzać w budowę typowego systemu komputerowego.
Dodatkowo podczas tych zajęć ukazywane będą zagadnienia związane z tworzeniem oprogramowania dla poszczególnych bloków omawianego systemu komputerowego.
Zajęcia laboratoryjne będą wprowadzeniem do tworzenia systemów komputerowych w ramach platform FPGA oraz z wykorzystaniem systemu operacyjnego Linux i prostych zasobów peryferyjnych na takich platformach.
W ramach zadania projektowego zakłada się powstanie systemu komputerowego z nieco bardziej zaawansowanymi układami peryferyjnymi. Dodatkowo w ramach zadania projektowego wymagane będzie utworzenie i zademonstrowanie działania tych peryferii.

Wymiar w jednostkach ECTS: 4 pkt.

Liczba godzin pracy studenta związanych z osiągnięciem efektów kształcenia (opis):

1. liczba godzin kontaktowych – 65 godz., w tym

obecność na wykładach	30 godz.,
obecność na zajęciach laboratoryjnych	15 godz.,
obecność na zajęciach projektowych	15 godz.,
konsultacje	5 godz.

2. praca własna studenta – 40 godz., w tym
   przygotowanie do zajęć laboratoryjnych (rozwiązanie odpowiedniej liczby zadań) 15 godz.,
   przygotowanie do zajęć projektowych (zapoznanie się z literaturą) i realizacja projektu 25 godz.,

Łączny nakład pracy studenta wynosi 108 godz., **co odpowiada** 4 pkt. **ECTS.**

Liczba punktów ECTS na zajęciach wymagających bezpośredniego udziału nauczycieli akademickich: 2,5 pkt. ECTS, co odpowiada 65 godz. kontaktowym.

Liczba punktów ECTS, którą student uzyskuje w ramach zajęć o charakterze praktycznym: 2.5 pkt. ECTS, co odpowiada 70 godz. zajęć laboratoryjnych i projektowych oraz przygotowaniu do tych zajęć

Efekty uczenia się/kształcenia:

efekty kształcenia/uczenia się	forma zajęć/ technika kształcenia	sposób weryfikacji (oceny)*	odniesienie do efektów uczenia się dla programu
Wiedza			
w1: ma wiedzę niezbędną do zrozumienia: budowy i działania mikroprocesora i mikrokontrolera, budowy programów wykonywanych bezpośrednio przez mikroprocesor, sposobu reprezentowania podstawowych danych przez mikroprocesor	Wykład + laboratorium + projekt	laboratorium + projekt	W04
w2: ma wiedzę niezbędną do zrozumienia procesu kompilacji i budowy kodu tworzonego w języku C/C++, posiada wiedzę niezbędną do uruchamiania oprogramowania w języku C/C++	Wykład + laboratorium + projekt	Kolokwium + laboratorium + projekt	W05
w3: ma wiedzę jak zbudowane są pamięci dla systemów komputerowych, zna ich zalety i wady	Wykład	Kolokwium	W04
w4: ma wiedzę niezbędną do zrozumienia: budowy styku systemu komputerowego z otoczeniem, działania mechanizmu przerwań w systemach komputerowych, działania zaawansowanych układów peryferyjnych i magistral łączących je z mikroprocesorem, w szczególności zna techniki podnoszenia bezpieczeństwa takich systemów	Wykład + laboratorium + projekt	Kolokwium + laboratorium + projekt	W04 W07
w5: ma wiedzę o cyklu życia systemu komputerowego, roli i działaniu programów ładujących system operacyjny oraz podstawach współpracy systemu operacyjnego z warstwą sprzętową.	Wykład + laboratorium + projekt	Kolokwium + laboratorium + projekt	W04 W08 W07
w6: ma wiedzę na temat budowy niskopoziomowych mechanizmów procesora wspierania systemów operacyjnych dla podniesienia bezpieczeństwa	Wykład	Kolokwium	W07 W04
Umiejętności			
u1: potrafi (począwszy od schematu ideowego) zbudować system komputerowy, rozszerzyć go o nowe moduły pamięci i	Wykład + laboratorium + projekt	Laboratorium + projekt	U03 U01 U08

układy peryferyjne oraz zbudować optymalny i odpowiedni dla takiego systemu dekoder adresów			
u2: potrafi symulacyjnie zweryfikować działanie zbudowanego przez siebie systemu komputerowego	Laboratorium + projekt	Laboratorium + projekt	U03
u3: potrafi do układów peryferyjnych zbudowanego przez siebie systemu komputerowego dodać podsystem generacji przerwań	Wykład + laboratorium + projekt	Laboratorium + projekt	U03 U01 U08
u4: potrafi napisać program obsługi przerwań dla zbudowanego przez siebie systemu komputerowego a następnie zweryfikować praktycznie jego działanie	Laboratorium + projekt	Laboratorium + projekt	U03 U08
u5: potrafi w zespole opracować i zrealizować harmonogram prac związanych z zaprojektowaniem, uruchomieniem oraz przetestowaniem budowanego przez siebie systemu komputerowego w sposób zapewniający dotrzymanie terminów	Laboratorium + projekt	Laboratorium + projekt	U09
Kompetencje społeczne			
ks1: rozumie potrzebę stałego aktualizowania i wzbogacania posiadanej wiedzy – podnoszenia kompetencji zawodowych, osobistych i społecznych	Wykład + laboratorium + projekt	Laboratorium + projekt	KS01

Uwagi:

Data i podpis autora (kierownika zespołu autorskiego):