Architektury systemów komputerowych

Pracownia 2: "Architektura procesora MIPS"

Przed przystąpieniem do rozwiązywania zadań należy przeczytać:

- Harris & Harris: 6.1 – 6.7 (drugie wydanie!)

Do realizacji poniższych zadań użyj, jeśli zajdzie taka potrzeba, odpowiednich wywołań systemowych opisanych w dokumentacji symulatora MARS. Dodatkowo należy używać odpowiednich dyrektyw asemblera oraz pseudoinstrukcji, a dla własnej wygody makrodefinicji. Jeśli implementujesz funkcje to należy zadbać o zachowywanie / odtwarzanie wartości rejestrów zgodnie z konwencją MIPS ABI (porozdziały 6.4.6, 6.7.4).

Symulator jest dostępny tu: http://courses.missouristate.edu/kenvollmar/mars. Warto zapoznać się z dokumentacją przed przystąpieniem do rozwiązywania zadań – w szczególności z pełną listą instrukcji i makrodefinicjami.

Każde zadanie musi być odpowiednio przygotowane do prezentacji – tj. musi zawierać w sobie co najmniej jeden test pokazujący poprawność implementacji.

Zadanie 1

Napisz program, który zlicza w pętli ile jedynek znajduje się w bitowej reprezentacji liczby znajdującej się w rejestrze `$a0`, wynik ma znaleźć się w rejestrze `$v0`.

Zadanie 1b [bonus]

Powtórz poprzednie zadanie z tym, że rozwiązanie musi działać w $O(log_2 N)$ krokach, gdzie N jest ilością bitów w słowie maszynowym.

Zadanie 2

Napisz funkcję przyjmującą adres do ciągu znaków zakończonego znakiem `NUL` (`\0`) w rejestrze `$a0` i sprawdzającą czy ów ciąg jest palindromem – wynik (wartość logiczna `0` lub `1`) ma być zwrócona w `$v0`. Przetestuj swoją funkcję wczytując dane od użytkownika.

Zadanie 3

Zaimplementuj funkcję wyznaczającą n-ty wyraz ciągu Fibonacciego metodą rekurencyjną. Zadbaj o odpowiednie przechowywanie danych na stosie. Liczbę n wczytaj z klawiatury.

Zadanie 4

Napisz funkcję, która sortuje tablicę liczb całkowitych (słowa 32-bitowe), której adres jest podany jest w rejestrze `$a0`, a rozmiar w rejestrze `$a1`. Użyj metody sortowania przez wstawianie.

Zadanie 4b [bonus]

Powtórz poprzednie zadanie, ale tym razem zaimplementuj prostą wersję algorytmu szybkiego sortowania.

Zadanie 5

Napisz funkcję obliczającą metodą pisemną pierwiastek kwadratowy z nieujemnej liczby całkowitej podanej w rejestrze `$a0`. Wynika ma zostać zwrócony w `$v0`.

Zadanie 6

Używając koprocesora matematycznego (ang. *Floating Point Unit*) zaimplementuj funkcję, która oblicza $sin(x)$ używając rozwinięcia w szereg. Wartość x należy przekazywać `$fa0`, a wynik w zwracać w `$fv0`. Zanim zaczniesz obliczać wielomian upewnij się, że x jest w "dobrym" przedziale – inaczej rozwinięcie w szereg Taylor'a da zły wynik. Należy przestrzegać `ABI` dla rejestrów zmiennoprzecinkowych!

Zadanie 7

Załóżmy, że rejestry `$a0` i `$a1` przechowują wartości zmiennopozycyjne pojedynczej precyzji w standardzie `IEEE754`. Napisz program, który wymnoży te liczby i umieści wynik w rejestrze `$v0`. Wolno używać wyłącznie instrukcji na liczbach całkowitych! Do sprawdzenia poprawności wyniku można użyć wtyczki symulatora o nazwie *Floating Point Representation*.

Zadanie 8

Skonstruuj program, który w pętli będzie losował lewy górny i prawy dolny róg prostokąta oraz jego kolor, a następnie wrysowywał taki prostokąt do pamięci "ekranu". Należy uśrednić kolor (tj. poszczególne składowe RGB) rysowanego prostokąta z kolorem pikseli, na których rysujemy prostokąt. Do wyświetlenia ekranu używaj narzędzia "Bitmap Display", gdzie wyświetlany obszar ma mieć rozdzielczość 256x256 z pikselami 1x1.

Zadanie 9

Używając narzędzia *Digital Sim Lab* napisz grę "*ciepło – zimno*". Będzie ona polegać na tym, że komputer wylosuje pewną liczbę z zakresu 0 – 255. Zadaniem użytkownika będzie w ciągu 8 tur znaleźć wylosowaną liczbę. Gdy wpisana liczba jest zbyt mała na wyświetlaczu ma się pojawić napis `lo`, jeśli za duża to `hi`, a program powinien czekać na następną liczbę. Jeśli podana liczba jest równa lub przekroczona dozwoloną ilość kroków to przez kilka sekund mają mrugać kropki i gra ma się zrestartować.

Zadanie 10 [2pkt, bonus]

Procesory MIPS32 nie potrafią wczytywać / zapisywać słów znajdujących się pod adresami niepodzielnymi przez rozmiar słowa – tj. instrukcje dostępu wygenerują w takim przypadku wyjątek. Napisz procedurę obsługi wyjątku, która pozwoli zapisać dane pod wskazany adres rozbijając instrukcje odczytu i zapisu słowa na bajty (nie wolno korzystać z pseudoinstrukcji ULW / USW i podobnych). Wracając z przerwania przeskocz o jedną instrukcję do przodu, tak żeby stworzyć złudzenie, że błąd nie wystąpił, a wykonanie instrukcji się powiodło. Pokaż, że Twoje rozwiązanie działa dla instrukcji LW i SW dla dowolnych adresów. Więcej nt. przerwań można przeczytać w podzakładce *MIPS → Exceptions*.

Zadanie 11 [bonus]

Procesory MIPS32v2 posiadają przydatne instrukcje SEB (sign extend byte), EXT (extract bits), INS (insert bits), które nie występują w MIPS32v1. Semantykę wymienionych instrukcji można znaleźć w MIPS32® Instruction Set Quick Reference. Bez użycia pseudoinstrukcji i instrukcji skoków zaimplementuj w/w operacje z użyciem makr asemblerowych. Więcej nt. makr można przeczytać w podzakładce *MIPS → Macros*. Pamietaj, że makra nie mogą modyfikować żadnych rejestrów poza docelowym i $at oraz powinny unikać dostępów do pamięci.

Krystian Bacławski