Projekt Architektura Systemów Komputerowych 2

Wydział Elektrotechniki Automatyki i Informatyki

Politechnika Świętokrzyska

Studia: **Stacjonarne I stopnia**	Kierunek: **Informatyka**
Data oddania: **18.01.2017**	Grupa: **2ID15**
Ocena:	1. Monika Molenda 2. Bartłomiej Osak 3. Tomasz Pasternak

Temat projektu:

Projekt Nr 5

Polecenie:

Wykonać projekt mikroprocesora oraz układów towarzyszących zgodnie z przedstawionymi we wstępie do instrukcji specyfikacją i ograniczeniami. Ponadto mikroprocesor musi:

- mieć możliwość zaadresowania 4096 słów pamięci operacyjnej,
- wspierać adresowania: domyślne, natychmiastowe, indeksowe,
- wspierać segmentację pamięci z podziałem na segment kodu programu i segment danych,
- posiadać odpowiednią liczbę rejestrów segmentowych,
- posiadać rejestr licznika rozkazów (tylko do odczytu),
- posiadać 4 rejestry uniwersalne,
- wykonywać rozkazy:
  - przesyłanie danych rej-nat, rej-rej, rej-pam,
  - dodawanie/odejmowanie rej-nat, rej-rej,
  - blokowe przesyłanie danych w pamięci z użyciem licznika powtórzeń,
  - porównywanie rej-rej,
  - wykonywanie skoku bezwarunkowego do adresu podanego jako liczba lub rejestr,
  - wykonywanie skoków warunkowych, gdy większe, mniejsze, równe,
  - wyliczanie wartości funkcji logicznych dla rej-nat, rej-rej.

Podstawową długością słowa mikroprocesora jest 8 bitów. Rejestr znaczników musi być aktualizowany po wykonaniu odpowiednik rozkazów. Słowo rozkazu mikroprocesora MUSI posiadać zmienną długość. Długość słowa na magistrali danych mikroprocesora ma wynosić 8 bitów. W pamięci należy przygotować program, który będzie demonstrował możliwości mikroprocesora (treść pseudokodu wraz z treścią assemblera należy zamieścić w sprawozdaniu).

1. Rejestry:

Symbol	Kod rejestru	Opis rejestru
R_0	00	Rejestr uniwersalny
R_1	01	Rejestr uniwersalny, licznik powtórzeń operacji blokowych
R_2	10	Rejestr uniwersalny
R_3	11	Rejestr uniwersalny
Inne rejestry		
Symbol		**Opis rejestru**
ACC_{ALU}		Akumulator ALU
ACC_{NAT}		Akumulator wartości natychmiastowej
FLAGS		Rejestr znaczników
CS, DS		Rejestry segmentowe
SI, DI		Rejestry indeksowe
PC		Rejestr licznika programu

2. Projekt rozkazu:

Kod grupy	Kod rozkazu	Opis rozkazu
Operacje jednoargumentowe		
000	**M - kod podgrupy , CC - kod operacji, RR – kod rejestru**	
	dla M=0	**M CC -- [nat8]**
	0 00 -- [nat8]	jmp - skok bezwarunkowy
	0 01 -- [nat8]	jg - skok warunkowy, gdy >
	0 10 -- [nat8]	jl - skok warunkowy, gdy <
	0 11 -- [nat8]	je - skok warunkowy, gdy =
	dla M=1	**M CC RR**
	1 00 RR	jmp – skok bezwarunkowy
Operacje dwuargumentowe		
001	**M - kod podgrupy, CCCC - kod operacji, RR - kod rejestru I, RR -kod rejestru II, F – operacja z flagą**	
	dla M=0	**M CCCC --- RR RR F**
	0 0000 ---RR RR 0	add RR, RR
	0 0000 ---RR RR 1	adc RR, RR
	0 0001 ---RR RR 0	sub RR, RR
	0 0001 ---RR RR 1	sbb RR, RR
	0 0010 ---RR RR -	cmp RR, RR
	0 0011 ---RR RR -	and RR, RR
	0 0100 ---RR RR -	or RR, RR
	0 0101 ---RR RR -	xor RR, RR
	dla M=1	**M CCCC ---RR -- F [nat8]**
	1 0000 ---RR -- 0 [nat8]	add RR, [nat8]
	1 0000 ---RR -- 1 [nat8]	adc RR, [nat8]
	1 0001 ---RR -- 0 [nat8]	sub RR, [nat8]
	1 0001 ---RR -- 1 [nat8]	sbb RR, [nat8]
	1 0010 ---RR -- - [nat8]	and RR, [nat8]
	1 0011 ---RR -- - [nat8]	or RR, [nat8]
	1 0100 ---RR -- - [nat8]	xor RR, [nat8]
	1 0101 ---RR -- 1	sub RR, 1
Operacje przesyłania danych rej-rej, rej-nat, rej-pam		
010	**M - kod podgrupy, RR - kod rejestru I, RR -kod rejestru II**	
	dla M=0	**M RR RR**
	0 RR RR	mov RR, RR
011	**M - kod podgrupy, CC – kod operacji, RR - kod rejestru**	
	dla M=0	**M -- RR [nat8]**
	0 00 RR [nat8]	mov RR, [nat8]
	dla M=1	**M -- RR 00 dla: [SI] ,01 dla: [DI]**
	1 00 RR	mov RR, [SI]
	1 01 RR	mov RR, [DI]
Operacja blokowego przesyłania danych		
100	**M - kod podgrupy, CC – kod operacji, R1 - licznik**	
	0 00 --	rep movsb [SI], [DI]

3. Założenia dla kontrolera:

Sygnały 1b: ADR_0, ADR_3
Sygnały 2b: ADR_1, ADR_2
Sygnały 4b: ADR_4
Sygnały ładowania rejestrów: LD_{aa}, LD_{ACC_NAT}, LD_{ACC_ALU}, LD_{FLAGS}.

4. Mikroinstrukcje:

Lp.	Cykl	Mikroinstrukcja	Opis
1.	$R_{aa} = WE$		Ładowanie wartości z wejścia układu do rejestrów.
	t0:	$ADR_0=0$, $ADR_1=aa$	
	t1:	$LD_{aa}=1$	
2.	$R_{aa} = R_{aa} + R_{bb}$		Dodawanie wartości dwóch rejestrów i zapis wyniku do rejestru **(ADD)**
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$, $ADR_4=0$	
	t1:	$LD_{ACC\_ALU}=1$	
	t2:	$LD_{FLAGS}=1$, $ADR_0=1$, $ADR_1=aa$	
	t3:	$LD_{aa}=1$	
3.	$R_{aa} = R_{aa} + R_{bb}$		Dodawanie wartości dwóch rejestrów oraz flagi CF i zapis wyniku do rejestru **(ADC)**
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$, $ADR_4=1$	
	t1:	$LD_{ACC\_ALU}=1$	
	t2:	$LD_{FLAGS}=1$, $ADR_0=1$, $ADR_1=aa$	
	t3:	$LD_{aa}=1$	
4.	$R_{aa} = R_{aa} - R_{bb}$		Odejmowanie wartości dwóch rejestrów i zapis wyniku do rejestru **(SUB)**
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$, $ADR_4=2$	
	t1:	$LD_{ACC\_ALU}=1$	
	t2:	$LD_{FLAGS}=1$, $ADR_0=1$, $ADR_1=aa$	
	t3:	$LD_{aa}=1$	
5.	$R_{aa} = R_{aa} - R_{bb}$		Odejmowanie wartości dwóch rejestrów oraz flagi CF i zapis wyniku do rejestru **(SBB)**
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$, $ADR_4=3$	
	t1:	$LD_{ACC\_ALU}=1$	
	t2:	$LD_{FLAGS}=1$, $ADR_0=1$, $ADR_1=aa$	
	t3:	$LD_{aa}=1$	
6.	R_{aa} cmp R_{bb}		Porównywanie dwóch rejestrów i wynik zapisany w rejestrze flag
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$	
	t1:	$LD_{FLAGS}=1$	
7.	$R_{aa} = R_{aa}$ and R_{bb}		Iloczyn logiczny dwóch rejestrów i zapis wyniku do rejestru
	t0:	$ADR_1=aa$, $ADR_2=bb$, $ADR_3=0$, $ADR_4=5$	
	t1:	$LD_{ACC\_ALU}=1$, $ADR_0=1$, $ADR_1=aa$	
	t2:	$LD_{aa}=1$	

8.	**$R_{aa} = R_{aa}$ or R_{bb}**		Suma logiczna dwóch rejestrów i zapis wyniku do rejestru
	t0:	ADR_1=aa, ADR_2=bb, ADR_3=0, ADR_4=6	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	
9.	**$R_{aa} = R_{aa}$ xor R_{bb}**		Różnica symetryczna dwóch rejestrów i zapis wyniku do rejestru
	t0:	ADR_1=aa, ADR_2=bb, ADR_3=0, ADR_4=7	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	
10.	**$R_{aa} = R_{bb}$**		Przesyłanie zawartości danego rejestru do wskazanego rejestru
	t0:	ADR_1=bb, ADR_4=8	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	
11.	**$R_{aa} = R_{aa}$ + [nat8]**		Dodawanie wartości rejestru i [nat], zapis wartości do rejestru **(ADD)**
	t0:	ADR_1=aa, ADR_3=1, ADR_4=0	
	t1:	LD_{ACC_ALU}=1	
	t2:	LD_{FLAGS}=1, ADR_0=1, ADR_1=aa	
	t3:	LD_{aa}=1	
12.	**$R_{aa} = R_{aa}$ + [nat8]**		Dodawanie wartości rejestru i [nat] oraz flagi CF, zapis wartości do rejestru **(ADC)**
	t0:	ADR_1=aa, ADR_3=1, ADR_4=1	
	t1:	LD_{ACC_ALU}=1	
	t2:	LD_{FLAGS}=1, ADR_0=1, ADR_1=aa	
	t3:	LD_{aa}=1	
13.	**$R_{aa} = R_{aa}$ - [nat8]**		Odejmowanie wartości rejestru i [nat], zapis wartości do rejestru **(SUB)**
	t0:	ADR_1=aa, ADR_3=1, ADR_4=2	
	t1:	LD_{ACC_ALU}=1	
	t2:	LD_{FLAGS}=1, ADR_0=1, ADR_1=aa	
	t3:	LD_{aa}=1	
14.	**$R_{aa} = R_{aa}$ - [nat8]**		Odejmowanie wartości rejestru i [nat] oraz flagi CF, zapis wartości do rejestru **(SBB)**
	t0:	ADR_1=aa, ADR_3=1, ADR_4=3	
	t1:	LD_{ACC_ALU}=1	
	t2:	LD_{FLAGS}=1, ADR_0=1, ADR_1=aa	
	t3:	LD_{aa}=1	
15.	**$R_{aa} = R_{aa}$ and [nat8]**		Iloczyn logiczny wartości rejestru i [nat], zapis wyniku do rejestru
	t0:	ADR_1=aa, ADR_3=1, ADR_4=5	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	

16.	**$R_{aa} = R_{aa}$ or [nat8]**		Suma logiczna wartości rejestru i [nat], zapis wyniku do rejestru
	t0:	ADR_1=aa, ADR_3=1, ADR_4=6	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	
17.	**$R_{aa} = R_{aa}$ xor [nat8]**		Różnica symetryczna wartości rejestru i [nat], zapis wyniku do rej
	t0:	ADR_1=aa, ADR_3=1, ADR_4=7	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=aa	
	t2:	LD_{aa}=1	
18.	**$ACC_{NAT} = WE$**		Ładowanie wartości z wejścia układu do rejestru ACC_{NAT}.
	t0:	ADR_0=0	
	t1:	LD_{ACC_NAT}=1	
19.	**$R_{01} = R_{01} - 1$**		Dekrementacja rejestru licznikowego w operacji blokowej
	t0:	ADR_1=1, ADR_4=10	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=1	
	t2:	LD_1=1	
20.	**R_2[DI] = R_3[SI]**		Przesyłanie blokowe bajtu spod offsetu adresu z SI(tu:R3) do komórki pod adresem w DI(tu:R2).
	t0:	ADR_1=3, ADR_4=8	
	t1:	LD_{ACC_ALU}=1, ADR_0=1, ADR_1=2	
	t2:	LD_2=1	

5. Format mikrooperacji

Operacja	ADR_0	ADR_1	ADR_2	ADR_3	ADR_4	LD_{aa}	LD_{ACC_NAT}	LD_{ACC_ALU}	LD_{FLAGS}
$R_{aa=}WE$	0	aa	-	-	-	-	-	-	-
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}+R_{bb}$	-	aa	bb	0	0000	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa} = R_{aa}+R_{bb}$ z CF	-	aa	bb	0	0001	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa} = R_{aa} - R_{bb}$	-	aa	bb	0	0010	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa} = R_{aa} - R_{bb}$ z CF	-	aa	bb	0	0011	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-

R_{aa} cmp R_{bb}	-	aa	bb	0	-	-	-	-	-
	-	-	-	-	-	-	-	-	1
$R_{aa}=R_{aa}$ and R_{bb}	-	aa	bb	0	0101	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$ or R_{bb}	-	aa	bb	0	0110	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$ xor R_{bb}	-	aa	bb	0	0111	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$R_{aa} = R_{bb}$	-	bb	-	-	1000	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$ACC_{NAT} = WE$	0	-	-	-	-	-	-	-	-
	-	-	-	-	-	-	-	1	-
$R_{aa}=R_{aa}$+[nat8]	-	aa	-	1	0000	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$+[nat8] z CF	-	aa	-	1	0001	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$-[nat8]	-	aa	-	1	0010	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$-[nat8] z CF	-	aa	-	1	0011	-	-	-	-
	-	-	-	-	-	-	-	1	-
	1	aa	-	-	-	-	-	-	1
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$ and [nat8]	-	aa	-	1	0101	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$ or [nat8]	-	aa	-	1	0110	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-
$R_{aa}=R_{aa}$ xor [nat8]	-	aa	-	1	0111	-	-	-	-
	1	aa	-	-	-	-	-	1	-
	-	-	-	-	-	1	-	-	-

$R_{01}= R_{01}-1$	-	01	-	-	1010	-	-	-	-
	1	01	-	-	-	-	-	1	-
	-	-	-	-	-	$1_{(01)}$	-	-	-
$R_2[DI]=R_3[SI]$	-	11	-	-	1000	-	-	-	-
	1	10	-	-	-	-	-	1	-
	-	-	-	-	-	$1_{(10)}$	-	-	-