|  |  |
| --- | --- |
| **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** |
| R­0 | 00 | Rejestr uniwersalny |
| R­1 | 01 | Rejestr uniwersalny, licznik powtórzeń operacji blokowych |
| R­2 | 10 | Rejestr uniwersalny |
| R­3 | 11 | Rejestr uniwersalny |
| **Inne rejestry** | | |
| **Symbol** | | **Opis rejestru** |
| ACCALU | | Akumulator ALU |
| ACCNAT | | Akumulator wartości natychmiastowej |
| FLAGS | | Rejestr znaczników |
| CS, DS | | Rejestry segmentowe |
| SI, DI | | Rejestry indeksowe |
| PC | | Rejestr licznika programu |

1. **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] | |

1. **Założenia dla kontrolera:**

**Sygnały 1b:** ADR0, ADR3

**Sygnały 2b:** ADR1, ADR2

**Sygnały 4b:** ADR4

**Sygnały ładowania rejestrów:** LDaa, LDACC\_NAT, LDACC\_ALU, LDFLAGS.

1. **Mikroinstrukcje:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Cykl** | **Mikroinstrukcja** | **Opis** |
| **1.** | **Raa = WE** | | Ładowanie wartości z wejścia układu do rejestrów. |
|  | t0: | ADR0=0, ADR1=aa |  |
| t1: | LDaa=1 |
| **2.** | **Raa = Raa + Rbb** | | Dodawanie wartości dwóch rejestrów i zapis wyniku do rejestru **(ADD)** |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=0 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **3.** | **Raa = Raa + Rbb** | | Dodawanie wartości dwóch rejestrów oraz flagi CF i zapis wyniku do rejestru  **(ADC)** |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=1 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **4.** | **Raa = Raa - Rbb** | | Odejmowanie wartości dwóch rejestrów i zapis wyniku do rejestru  **(SUB)** |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=2 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **5.** | **Raa = Raa - Rbb** | | Odejmowanie wartości dwóch rejestrów oraz flagi CF i zapis wyniku do rejestru  **(SBB)** |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=3 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **6.** | **Raa cmp Rbb** | | Porównywanie dwóch rejestrów i wynik zapisany w rejestrze flag |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0 |  |
| t1: | LDFLAGS=1 |
| **7.** | **Raa = Raa and Rbb** | | Iloczyn logiczny dwóch rejestrów i zapis wyniku do rejestru |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=5 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **8.** | **Raa = Raa or Rbb** | | Suma logiczna dwóch rejestrów i zapis wyniku do rejestru |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=6 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **9.** | **Raa = Raa xor Rbb** | | Różnica symetryczna dwóch rejestrów i zapis wyniku do rejestru |
|  | t0: | ADR1=aa, ADR2=bb, ADR3=0, ADR4=7 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **10.** | **Raa = Rbb** | | Przesyłanie zawartości danego rejestru do wskazanego rejestru |
|  | t0: | ADR1=bb, ADR4=8 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **11.** | **Raa = Raa + [nat8]** | | Dodawanie wartości rejestru i [nat], zapis wartości do rejestru  **(ADD)** |
|  | t0: | ADR1=aa, ADR3=1, ADR4=0 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **12.** | **Raa = Raa + [nat8]** | | Dodawanie wartości rejestru i [nat] oraz flagi CF, zapis wartości do rejestru  **(ADC)** |
|  | t0: | ADR1=aa, ADR3=1, ADR4=1 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **13.** | **Raa = Raa - [nat8]** | | Odejmowanie wartości rejestru i [nat], zapis wartości do rejestru  **(SUB)** |
|  | t0: | ADR1=aa, ADR3=1, ADR4=2 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **14.** | **Raa = Raa - [nat8]** | | Odejmowanie wartości rejestru i [nat] oraz flagi CF, zapis wartości do rejestru  **(SBB)** |
|  | t0: | ADR1=aa, ADR3=1, ADR4=3 |  |
| t1: | LDACC\_ALU=1 |
| t2: | LDFLAGS=1, ADR0=1, ADR1=aa |
| t3: | LDaa=1 |
| **15.** | **Raa = Raa and [nat8]** | | Iloczyn logiczny wartości rejestru i [nat], zapis wyniku do rejestru |
|  | t0: | ADR1=aa, ADR3=1, ADR4=5 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **16.** | **Raa = Raa or [nat8]** | | Suma logiczna wartości rejestru i [nat], zapis wyniku do rejestru |
|  | t0: | ADR1=aa, ADR3=1, ADR4=6 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |
| t2: | LDaa=1 |
| **17.** | **Raa = Raa xor [nat8]** | | Różnica symetryczna wartości rejestru i [nat], zapis wyniku do rej |
|  | t0: | ADR1=aa, ADR3=1, ADR4=7 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=aa |  |
| t2: | LDaa=1 |
| **18.** | **ACCNAT = WE** | | Ładowanie wartości z wejścia układu do rejestru ACCNAT. |
|  | t0: | ADR0=0 |  |
| t1: | LDACC\_NAT=1 |
| **19.** | **R01 = R01 - 1** | | Dekrementacja rejestru licznikowego w operacji blokowej |
|  | t0: | ADR1=1, ADR4=10 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=1 |  |
| t2: | LD1=1 |
| **20.** | **R2[DI] = R3[SI]** | | Przesyłanie blokowe bajtu spod offsetu adresu z SI(tu:R3) do komórki pod adresem w DI(tu:R2). |
|  | t0: | ADR1=3, ADR4=8 |  |
| t1: | LDACC\_ALU=1, ADR0=1, ADR1=2 |  |
| t2: | LD2=1 |

1. **Format mikrooperacji**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Operacja** | **ADR0** | **ADR1** | **ADR2** | **ADR3** | **ADR4** | **LDaa** | **LDACC\_NAT** | **LDACC\_ALU** | **LDFLAGS** |
| Raa=WE | 0 | aa | - | - | - | - | - | - | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa+Rbb | - | aa | bb | 0 | 0000 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa = Raa+Rbb z CF | - | aa | bb | 0 | 0001 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa = Raa - Rbb | - | aa | bb | 0 | 0010 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa = Raa - Rbb z CF | - | aa | bb | 0 | 0011 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa cmp Rbb | - | aa | bb | 0 | - | - | - | - | - |
| - | - | - | - | - | - | - | - | 1 |
| Raa=Raa and Rbb | - | aa | bb | 0 | 0101 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa or Rbb | - | aa | bb | 0 | 0110 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa xor Rbb | - | aa | bb | 0 | 0111 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa = Rbb | - | bb | - | - | 1000 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| ACCNAT = WE | 0 | - | - | - | - | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| Raa=Raa+[nat8] | - | aa | - | 1 | 0000 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa+[nat8] z CF | - | aa | - | 1 | 0001 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa-[nat8] | - | aa | - | 1 | 0010 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa-[nat8] z CF | - | aa | - | 1 | 0011 | - | - | - | - |
| - | - | - | - | - | - | - | 1 | - |
| 1 | aa | - | - | - | - | - | - | 1 |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa and [nat8] | - | aa | - | 1 | 0101 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa or [nat8] | - | aa | - | 1 | 0110 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| Raa=Raa xor [nat8] | - | aa | - | 1 | 0111 | - | - | - | - |
| 1 | aa | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1 | - | - | - |
| R01= R01-1 | - | 01 | - | - | 1010 | - | - | - | - |
| 1 | 01 | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1(01) | - | - | - |
| R2[DI]=R3[SI] | - | 11 | - | - | 1000 | - | - | - | - |
| 1 | 10 | - | - | - | - | - | 1 | - |
| - | - | - | - | - | 1(10) | - | - | - |