|  |
| --- |
| СПБНИУ ИТМО |
| Курсовая работа |
| Курс: Организация ЭВМ и систем |
|  |
| **Артем Кудряшов** |
| **08.05.2014** |

|  |
| --- |
|  |

## Задание:

Вариант №7: dec {ri,@rj,ad} anl c,{bit,/bit} mov a,{ri,#d} jz rel

## Этап 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда ANL <байт-назначения>,<байт-источникa>Команда "логическое "И" для переменных-байтов" выполняет операцию логического "И" над битами указанных перемнных и помещает результат в байт-назначения. Эта операция не влияет на состояние флагов.Для операнда обеспечивают следующие комбинации шести режимов адресации:байтом назначения является аккумулятор (А):1. регистровый
2. прямой
3. косвенно-регистровый
4. непосредственный

байтом назначения является прямой адрес (direct):1. прямой аккумуляторный
2. непосредственный (байт-источник равен константе)

Рассмотрим их.

|  |  |
| --- | --- |
| Ассемблер: | ANL A,Rn ; где n=0-7 |
| Код: |

|  |
| --- |
| 0 1 0 1 1 rrr |

 | , где rrr=000-111 |
| Время: | 1 цикл |
| Алгоритм: | (A) : = (A) AND (Rn) |
| Пример: |  ;(A)=FEH, (R2)=C5HANL A,R2 ;(A)=C4H, (R2)=C5H  |

|  |  |
| --- | --- |
| Ассемблер: | ANL A,<direct> |
| Код: |

|  |
| --- |
| 0 1 0 1 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |
| Время: | 1 цикл |
| Алгоритм: | (A) : =(A) AND (direct) |
| Пример: |  ;(A)=A3H, (PSW)=86HANL A,PSW ;(A)=82H, (PSW)=86H  |

|  |  |
| --- | --- |
| Ассемблер: | ANL A,@Ri ; где i=0,1 |
| Код: |

|  |
| --- |
| 0 1 0 1 0 1 1 i |

 | , где i=0,1 |
| Время: | 1 цикл |
| Алгоритм: | (A) : =(A) AND (Ri) |
| Пример: |  ;(A)=BCH, (ОЗУ [35])=47H, (R0)=35H, ANL A,@R0 ;(A)=04H, (ОЗУ [35])=47H |

|  |  |
| --- | --- |
| Ассемблер: | ANL A, #data |
| Код: |

|  |
| --- |
| 0 1 0 1 0 1 0 0 |

 |

|  |
| --- |
| #data8 |

 |
| Время: | 1 цикл |
| Алгоритм: | (A) : = (A)AND #data |
| Пример: |  ;(A)=36HANL A,#0DDH ;(A)=14H  |

|  |  |
| --- | --- |
| Ассемблер: | ANL <direct>, A |
| Код: |

|  |
| --- |
| 0 1 0 1 0 0 1 0 |

 |

|  |
| --- |
| direct address |

 |
| Время: | 1 цикл |
| Алгоритм: | (direct) : = (direct) AND (A) |
| Пример: |  ;(A)=55H, (P2)=AAHANL P2,A ;(P2)=00H, (A)=55H  |

|  |  |
| --- | --- |
| Ассемблер: | ANL <direct>, #data |
| Код: |

|  |
| --- |
| 0 1 0 1 0 0 1 1 |

 |

|  |
| --- |
| direct address |

 |

|  |
| --- |
| #data8 |

 |
| Время: | 2 циклa |
| Алгоритм: | (direct) : = (direct) AND #data |  |
| Пример: |  ;(P1)=FFHANL P1,#73H ;(P1)=73H |  |

 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда DEC <байт>Команда "декремент" производит вычитание "1" из указанного операнда. Начальное значение 00Н перейдет в 0FFH. Команда DEC не влияет на флаги. Этой командой допускается четыре режима адресации операнда:1. к аккумулятору
2. регистровый
3. прямой
4. косвенно-регистровый

Рассмотрим их.

|  |  |
| --- | --- |
| **Ассемблер:** | DEC A |
| **Код:** |

|  |
| --- |
| 0 0 0 1 0 1 0 0 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (A) : =(A)-1 |
| **Пример:** |  ;(A)=11H, (C)=1, (AC)=1DEC A ;(A)=10H, (C)=1, (AC)=1  |

|  |  |
| --- | --- |
| **Ассемблер:** | DEC Rn ; где n=0-7 |
| **Код:** |

|  |
| --- |
| 0 0 0 1 1 rrr |

 | где rrr=000-111 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (Rn) :=(Rn)-1 |
| **Пример:** |  ;(R1)=7FH, ;(ОЗУ[7F])=40H, (ОЗУ[7F])=00H DEC @R1  DEC R1 DEC @R1 ;(R1)=7EH, ;(ОЗУ[7F])=3FH, (ОЗУ[7F])=FFH   |

|  |  |
| --- | --- |
| **Ассемблер:** | DEC <direct> |
| **Код:** |

|  |
| --- |
| 0 0 0 1 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (direct) : = (direct)-1 |
| **Пример:** |  ;(SCON)=A0H, (C)=1, (AC)=1DEC SCON ;(SCON)=9FH, (C)=1, (AC)=1 |

|  |  |
| --- | --- |
| **Ассемблер:** | DEC @Ri ; где i=0,1 |
| **Код:** |

|  |
| --- |
| 0 0 0 1 0 1 1 i |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | ((Ri) :=((Ri)-1) |
| **Пример:** |  ;(R1)=7FH, ;(ОЗУ[7F])=40H, (ОЗУ[7F])=00H DEC @R1 DEC R1DEC @R1 ;(R1)=7EH, ;(ОЗУ[7F])=3FH, (ОЗУ[7F])=FFH  |

*Примечание.* Если эта команда используется для изменения информации на выходе порта, значение, используемое как исходные данные, считывается из "защелки" порта, а не с выводов БИС. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Команда "переслать переменную-байт" пересылает переменную-байт, указанную во втором операнде, в ячейку, указанную в первом операнде. Содержимое байта источника не изменяется. Эта команда на флаги и другие регистры не влияет. Команда "MOV" допускает 15 комбинаций адресации байта-источника и байта-назначения.

|  |  |
| --- | --- |
| **Ассемблер:** | MOV A,Rn; где n=0-7 |
| **Код:** |

|  |
| --- |
| 1 1 1 0 1 rrr |

 |

|  |
| --- |
| где rrr=000-111 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (A) : =(Rn) |
| **Пример:** |  ;(A)=FAH, (R4)=93HMOV A,R4 ;(A)=93H, (R4)=93H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV A, <direct> |
| **Код:** |

|  |
| --- |
| 1 1 1 0 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (A) :=(direct) |
| **Пример:** |  ;(A)=93H, (ОЗУ[40])=10H, (R0)=40HMOV A,40H ;(A)=10H, (ОЗУ[40])=10H, (R0)=40H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV A,@Ri; где i=0,1 |
| **Код:** |

|  |
| --- |
| 1 1 1 0 0 1 1 i |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (A) : = ((Ri)) |
| **Пример:** |  ;(A)=10H, (R0)=41H, (ОЗУ[41])=0CAHMOV A,@R0 ;(A)=CAH, (R0)=41H, (ОЗУ[41])=0CAH |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV A,#data |
| **Код:** |

|  |
| --- |
| 0 1 1 1 0 1 0 0 |

 |

|  |
| --- |
| #data8 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (A) :=<#data8> |
| **Пример:** |  ;(A)=C9H (11001001B)MOV A,#37H ;(A)=37H (00110111B)  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV Rn ,A; где n=0-7 |
| **Код:** |

|  |
| --- |
| 1 1 1 1 1 rrr |

 |

|  |
| --- |
| где rrr=000-111 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (Rn) : =(A) |
| **Пример:** |  ;(A)=38H, (R0)=42HMOV R0,A ;(A)=38H, (R0)=38H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV Rn, <direct>; где n=0-7 |
| **Код:** |

|  |
| --- |
| 1 0 1 0 1 rrr |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (Rn) :=(direct) |
| **Пример:** |  ;(R0)=39H, (P2)=0F2HMOV R0,P2 ;(R0)=F2H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV Rn,#data; где n=0-7 |
| **Код:** |

|  |
| --- |
| 0 1 1 1 1 rrr |

 |

|  |
| --- |
| #data8 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (Rn) :=<#data8> |
| **Пример:** |  ;(R0)=0F5HMOV R0,#49H ;(R0)=49H   |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV <direct>,A |
| **Код:** |

|  |
| --- |
| 1 1 1 1 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | (direct) :=(A) |
| **Пример:** |  ;(P0)=FFH, (A)=4BHMOV P0,A ;(P0)=4BH, (A)=4BH  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV <direct>, Rn ; где n=0-7 |
| **Код:** |

|  |
| --- |
| 1 0 0 0 1 rrr |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (direct) :=(Rn) |
| **Пример:** |  ;(PSW)=C2H, (R7)=57HMOV PSW,R7 ;(PSW)=57H, (R7)=57H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV <direct>, <direct> |
| **Код:** |

|  |
| --- |
| 1 0 0 0 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (direct) :=(direct) |
| **Пример:** |  ;(ОЗУ[45])=33H, (ОЗУ[48])=0DEHMOV 48H,45H ;(ОЗУ[45])=33H, (ОЗУ[45])=33H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV <direct>,@Ri ; где i=0,1 |
| **Код:** |

|  |
| --- |
| 1 0 0 0 0 1 1 i |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (direct) :=((Ri)) |
| **Пример:** |  ;(R1)=49H, (ОЗУ[49])=0E3HMOV 51H,@R1 ;(ОЗУ[51])=0E3H, (ОЗУ[49])=0E3H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV <direct>, #data |
| **Код:** |

|  |
| --- |
| 0 1 1 1 0 1 0 1 |

 |

|  |
| --- |
| direct address |

 |

|  |
| --- |
| #data8 |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (direct) :=<#data8> |
| **Пример:** |  ;(ОЗУ[5F])=9BHMOV 5FH,#07H ;(ОЗУ[5F])=07H  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV @Ri,A; где i=0-7 |
| **Код:** |

|  |
| --- |
| 1 1 1 1 0 1 1 i |

 |

|  |
| --- |
| где i=0,1 |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | ((Ri)) : =(A) |
| **Пример:** |  ;(R1)=51H, (ОЗУ[48])=75H, (A)=0BDHMOV @R1,A ;(ОЗУ[48])=0BDH  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV @Ri, <direct>, где i=0,1 |
| **Код:** |

|  |
| --- |
| 1 0 1 0 0 1 1 i |

 |

|  |
| --- |
| direct address |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | ((Ri)) :=(direct) |
| **Пример:** |  ;(R0)=51H, (ОЗУ[51])=0E3H, (P0)=0ACHMOV @R0,P0 ;(A)=10H, (ОЗУ[51])=0ACH  |

|  |  |
| --- | --- |
| **Ассемблер:** | MOV Ri,#data ; где i=0,1 |
| **Код:** |

|  |
| --- |
| 0 1 1 1 0 1 1 i |

 |  |
| **Время:** | 1 цикл |
| **Алгоритм:** | ((Ri)) : =<#data8> |
| **Пример:** |  ;(ОЗУ[7E])=67H, (R1)=7EHMOV @R1,#0A9H ;(ОЗУ[7E])=0A9H, (R1)=7EH  |

 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Команда JZ <re18>**Команда "переход, если содержимое аккумулятора равно 0" выполняет ветвление по адресу, если все биты аккумулятора равны "0", в противном случае выполняется следующая команда. Адрес ветвления вычисляется сложением относительного смещения со знаком во втором байте команды (re18) и содержимым счетчика команд после прибавления к нему 2. Содержимое аккумулятора не изменяется. Эта команда на флаги не влияет.

|  |  |
| --- | --- |
| **Ассемблер:** | JZ <метка> |
| **Код:** |

|  |
| --- |
| 0 1 1 0 0 0 0 0 |

 |

|  |
| --- |
| re18 |

 |  |
| **Время:** | 2 циклa |
| **Алгоритм:** | (PC):=(PC)+2если (A)0, то (PC):=(PC)+<re18> |
| **Пример:** |  ;(A)=01H JZ LAB16 ;нет перехода на LAB16 DEC A LAB16: JZ LAB17 ;переход на метку LAB17 ...LAB17: CLR A  |

 |

## Этап 2,4

using System;

using System.Collections.Generic;

using System.Globalization;

namespace EVMKurs

{

 class Program

 {

 #region vars

 /// <summary>

 /// Аккумулятор

 /// </summary>

 readonly static Register Akk = new Register(16);

 /// <summary>

 /// Регистр команд

 /// </summary>

 readonly static Register CommandReg = new Register(16);

 /// <summary>

 /// Память

 /// </summary>

 readonly static MemmoryCell[] Memmory = new MemmoryCell[1024];

 #endregion

 #region Anl

 /// <summary>

 /// Метод - эмулятор команды ANL

 /// </summary>

 /// <param name="a">Регистр назначения</param>

 /// <param name="b">Регистр - участник операции</param>

 /// <param name="isRef">Флаг косвенной адресации</param>

 private static void Anl(Register a, Register b, bool isRef = false)

 {

 Anl(a, !isRef ? b.Value : Memmory[b.Value].Value);

 }

 /// <summary>

 /// Метод - эмулятор команды ANL

 /// </summary>

 /// <param name="a">Регистр назначения</param>

 /// <param name="b">Значение-участник или значение-адрес(в зависимости от isRef)</param>

 /// <param name="isRef">Флаг является ли адресация косвенной</param>

 private static void Anl(Register a, int b, bool isRef = false)

 {

 if (!isRef)

 {

 a.Value = a.Value | b;

 }

 else

 {

 a.Value = a.Value | Memmory[b].Value;

 }

 }

 #endregion

 #region JZ

 /// <summary>

 /// Метод - эмулятор команды JZ

 /// </summary>

 /// <param name="a">Адрес перехода</param>

 private static void Jz(int a)

 {

 if (Akk.Value == 0)

 CommandReg.Value = a;

 }

 #endregion

 #region DEC

 /// <summary>

 /// Метод-эмулятор команды DEC

 /// </summary>

 /// <param name="a">Регистр-участник</param>

 /// <param name="isRef">Флаг косвенной адресации</param>

 private static void Dec(Register a, bool isRef = false)

 {

 if (!isRef)

 {

 a--;

 }

 else

 {

 Dec(a.Value);

 }

 }

 /// <summary>

 /// Метод-эмулятор команды DEC

 /// </summary>

 /// <param name="a">Адрес ячейки-участницы</param>

 private static void Dec(int a)

 {

 Memmory[a]--;

 }

 #endregion

 #region MOV

 /// <summary>

 /// Метод - эмулятор команды Mov

 /// </summary>

 /// <param name="a">Регистр назначения</param>

 /// <param name="b">Регистр - участник операции</param>

 /// <param name="isARef">Флаг косвенной адресации регистра а</param>

 /// <param name="isBRef">Флаг косвенной адресации регистра b</param>

 private static void Mov(Register a, Register b, bool isARef = false, bool isBRef = false)

 {

 if (!isARef)

 {

 a.Value = !isBRef ? b.Value : Memmory[b.Value].Value;

 return;

 }

 Memmory[a.Value].Value = !isBRef ? b.Value : Memmory[b.Value].Value;

 }

 /// <summary>

 /// Метод - эмулятор команды Mov

 /// </summary>

 /// <param name="a">Регистр назначения</param>

 /// <param name="b">Константа - адрес, или константа-значение (в зависимости от isBRef)</param>

 /// <param name="isARef">Флаг косвенной адресации регистра а</param>

 /// <param name="isBRef">Флаг косвенной адресации b</param>

 private static void Mov(Register a, int b, bool isARef = false, bool isBRef = false)

 {

 if (!isARef)

 {

 a.Value = !isBRef ? b : Memmory[b].Value;

 return;

 }

 Memmory[a.Value].Value = !isBRef ? b : Memmory[b].Value;

 }

 #endregion

 static void Main(string[] args)

 {

 Mov(Akk, Akk, isBRef: true);

 Akk.Value = 10;

 Console.WriteLine(Akk.Value + "\n");

 Console.WriteLine(Akk.GetValueWithCounting(2) + "\n\n");

 Akk.Value = -10;

 Console.WriteLine(Akk.GetValueWithCounting(2));

 Console.ReadLine();

 Akk.Value = 2;

 Dec(Akk);

 Console.WriteLine(Akk.Value);

 Akk.Value = 0;

 Dec(Akk);

 Console.WriteLine(Akk.GetValueWithCounting(16));

 Akk.Value = 1;

 Memmory[1] = new MemmoryCell(32, 10);

 Dec(Akk, true);

 Console.WriteLine(Memmory[1].Value);

 Dec(1);

 Console.WriteLine(Memmory[1].Value);

 Console.ReadLine();

 }

 }

 /// <summary>

 /// Класс, представляющий ячейку памяти

 /// </summary>

 internal class MemmoryCell

 {

 public static MemmoryCell operator --(MemmoryCell a)

 {

 if (a.Value == 0)

 a.Value = int.Parse(new string('f', (a.Capacity - 1) / 4), NumberStyles.AllowHexSpecifier);

 else

 a.Value--;

 return a;

 }

 private int \_value;

 /// <summary>

 /// Текущее значение регистра

 /// </summary>

 public int Value

 {

 get { return \_value; }

 set

 {

 if (Convert.ToString(value, 2).Length > Capacity)

 throw new Exception("Разрядность регистра меньше устанавливаемого занчения");

 \_value = value;

 }

 }

 /// <summary>

 /// Строковое представление значения регистра в соответствии с системой исчисления

 /// </summary>

 /// <param name="counting">Основание системы исчисления</param>

 /// <returns>Строковое представление значения регистра в соответствии с системой исчисления</returns>

 public string GetValueWithCounting(int counting)

 {

 var tempResult = Convert.ToString(Value, counting);

 var charCounter = Capacity / (int)Math.Log(counting, 2);

 return tempResult.Length < charCounter

 ? string.Format("{0}{1}", new String('0', charCounter - tempResult.Length), tempResult)

 : tempResult.Substring(tempResult.Length - charCounter);

 }

 /// <summary>

 /// Разрядность регистра

 /// </summary>

 public byte Capacity { get; set; }

 /// <summary>

 /// Конструктор класса Register

 /// </summary>

 /// <param name="capacity">Разрядность регистра(до 32)</param>

 public MemmoryCell(byte capacity)

 {

 if (capacity > 32)

 throw new ArgumentOutOfRangeException("capacity");

 Capacity = capacity;

 \_value = 0;

 }

 public MemmoryCell(byte capacity, int value)

 : this(capacity)

 {

 Value = value;

 }

 }

 /// <summary>

 /// Класс, представляющий регистр

 /// </summary>

 internal class Register : MemmoryCell

 {

 public Register(byte capacity)

 : base(capacity)

 {

 }

 public Register(byte capacity, int value)

 : this(capacity)

 {

 Value = value;

 }

 public static Register operator --(Register a)

 {

 if (a.Value == 0)

 a.Value = int.Parse(new string('f', (a.Capacity - 1) / 4), NumberStyles.AllowHexSpecifier);

 else

 a.Value--;

 return a;

 }

 }

 }

Этап 5

Реализация команды Dec на основе реверсивного счетчика

Логические элементы, использованные в схеме:

**Синхронный RS-триггер**





**T-триггер**





**И-ИЛИ**





**Реверсивный счетчик:**





**Схема моделирования:**

****

Проверка работы схемы (результат моделирования в логическом анализаторе)

Прямой ход (U-D = 0)



Обратный ход (U-D = 1)



Реализация реверсивного счетчика на основе двухступенчатого D-триггера (триггер с управлением по спаду, построенной по технологии Master-Slave)

Элементы, использованные в схеме:

Двухступенчатый D-триггер с инвертором:









Проверка работы схемы (результат моделирования в логическом анализаторе)

Прямой ход (U-D = 0)



Обратный ход (U-D = 1)



Реализация синхронного реверсивного счетчика с параллельным переносом на основе T-триггеров:



Проверка работы схемы (результат моделирования в логическом анализаторе)

Прямой ход (U-D = 0)



Обратный ход (U-D = 1)



Исследование различий времени задержки между асинхронным и синхронным с параллельным переносом счетчиками:

Для исследования используем комбинацию при которой происходит изменение выходных сигналов Q0,Q1,Q2 c 000 на 111 и обратно:

|  |  |  |
| --- | --- | --- |
|  | Асинхронный счетчик | Синхронный счетчик |
| По фронту | По спаду | По фронту | По спаду |
| Q0 | 44 ns | 59 ns | 44 ns | 59 ns |
| Q1 | 132 ns | 147 ns | 168 ns | 184 ns |
| Q2 | 221 ns | 236 ns | 184 ns | 198 ns |

Асинхронный по фронту:



Асинхронный по спаду:



Синхронный по фронту:



Синхронный по спаду:



Из временных диаграмм и таблицы можно сделать вывод, что использование синхронных счетчиков относительно асинхронных тем целесообразнее, чем больше разрядность счетчика

Этап 6

Реализация счетчика с использованием языка ANDL (на D-триггерах)

Счетчиками называются последовательные логические схемы для счета тактовых импульсов. В некоторых счетчиках реализован счет вперед и назад (реверсивные счетчики), в некоторые счетчики можно загружать данные, а также обнулять их. Счетчики обычно определяют как D-триггеры (DFF и DFFE) и используют операторы IF.

Ниже приведена реализация 16-битного загружаемого счетчика со сбросом.

SUBDESIGN kurs

(

clk, load, ena, clr, d[15..0] : INPUT;

q[15..0] : OUTPUT;

)

VARIABLE

count[15..0] : DFF;

BEGIN

count[].clk = clk;

count[].clrn = !clr;

IF load THEN

count[].d = d[];

ELSIF ena THEN

count[].d = count[].q + 1;

ELSE

count[].d = count[].q;

END IF;

q[] = count[];

END;

В данном коде в секции VARIABLE объявлены 16 D-триггеров и им присвоены имена от **count0** до **count15.** В операторе IF определяется значение, загружаемое в триггеры по фронту синхросигнала (например, если загрузка запускается VCC, то триггерам присваивается значение **d[** **]**).

Вывод:

В рамках выполнения курсовой работы были рассмотрены некоторые команды (dec {ri,@rj,ad} anl c,{bit,/bit} mov a,{ri,#d} jz rel), их характеристики и особенности, разработана их реализация на языке c#, а также разработана схема, позволяющая реализовывать функционал одной из команд (dec) на основе реверсивного счетчика, схема была создана путем редактирования в графическом редакторе, а также с использованием возможностей языка AHDL.