////////////////////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////////////////////

void \_\_fastcall // вызывается для завершения выполнения инструкции на АИУ i\_Proc

FinalizeExceptSET\_DetectReadyForRun\_AddToBuffer\_AttemptExecMax\_Instructions(int i\_Proc)

{

// устанавливаются флаги готовности у входных операндов иных инструкций, зависящих

// по входным операндам от результата выполнения данной на АИУ номер i\_Proc

char Set[\_SET\_LEN],

aResult[\_ID\_LEN],

aPredicat[\_ID\_LEN], // поле предиката

str[\_4096], tmp[\_256];

bool s\_isPredicat, // TRUE, если ВЫПОЛНИВШИЙСЯ оператор есть ПРЕДИКАТ

isPredicat, // TRUE, если ЗАВИСИМЫЙ оператор ПРЕДИКАТ

flagNot, // TRUE, если в ЗАВИСИМОМ операторе первый символ имени флага предиката "!" или "~"

flagPredicat, // TRUE, если в имени ЗАВИСИМОГО оператора есть переменная (ХХХ или !XXX или !XXX)

flagPredicatTRUE; // TRUE, если в ЗАВИСИМОМ операторе флаг предиката TRUE (с учётом isNot)

//

if( !Regim ) // не выполнять - закончить счет ---------------------------------

return;

//

int i\_Set = Mem\_Proc[i\_Proc].i\_Set; // на этом АИУ выполнялась инструкция номер i\_Set

//

REAL Result = Mem\_Proc[i\_Proc].Result; // значение результата выполненной операции

//

strcpy( aResult, Mem\_Proc[i\_Proc].aResult ); // запомнили адрес результата выполнения инструкции i\_Set\_Result

//

s\_isPredicat = is\_Predicat( Mem\_Instruction[i\_Set].Set ); // TRUE, если выполнившийся оператор суть ПРЕДИКАТ

//

Add\_toData( i\_Set, Mem\_Proc[i\_Proc].aResult, Result ); // добавить результат выполнившейся инструкции

//

t\_printf( "-I- %s(){1}: АИУ номер %d выполнило инструкцию #%d [%s] [%d/%d/%d тактов] -I-",

\_\_FUNC\_\_, i\_Proc, i\_Set, Line\_Set(i\_Set, 1),

Mem\_Proc[i\_Proc].tick\_Start, localTick, localTick - Mem\_Proc[i\_Proc].tick\_Start);

//

Vizu\_Data(); // визуализировать...

//

////////////////////////////////////////////////////////////////////////////////

// добавили запись в набор строк Tpr для анализа загруженности АИУ...........

snprintf(str,sizeof(str), "%10d%10d%10d%10d%10d [%s]",

i\_Proc,

Mem\_Proc[i\_Proc].tick\_Start, localTick, localTick - Mem\_Proc[i\_Proc].tick\_Start,

i\_Set, Line\_Set(i\_Set, 1));

mTpr->Add(str); // добавили строку

//

////////////////////////////////////////////////////////////////////////////////

//

strcpy(str, ""); // очистим str

//

//==============================================================================

for(UI i=0; i<Really\_Set; i++) // по всем инструкциям из Mem\_Instruction[] ============

{

//

if( i == i\_Set ) // кроме только что выполненной.............................

continue;

//

strcpy( Set, Mem\_Instruction[i].Set ); // ... так удобнее для дальнейшей работы !

if( is\_SET(Set) ) // инструкция SET уже давно едИножды выполнена !

continue;

//

strcpy( aPredicat, Mem\_Instruction[i].aPredicat ); // будем работать с aPredicat, не изменяя Mem\_Instruction[i].aPredicat

//

isPredicat = is\_Predicat( Set ); // TRUE, если это инструкция ПРЕДИКАТ

//

////////////////////////////////////////////////////////////////////////////////

//

//--- проверяем, начинается ли имя переменной предиктора с '!' или '~'

if ( !isPredicat ) // это инструкция - НЕ предиктор

flagNot = ( aPredicat[0] == symbolNot\_1 || aPredicat[0] == symbolNot\_2 )

? TRUE : FALSE; // TRUE, если в поле aPredicat первый символ '!' или '~'

//

//--- проверяем, совпадает ли имя возврашённой переменной с именем -------------

//--- переменной в поле предиктора i-той инструкции ----------------------------

flagPredicat = FALSE; // начальная установка

if( !isPredicat && // это инструкция - НЕ предиктор

strcmp( aPredicat, trueLowerCase ) && // И это НЕ статический true

strcmp( aPredicat, falseLowerCase ) ) // И это НЕ статический false

{

if( flagNot ) // имя начинается с '!' или '~'

if( !strcmp( &aPredicat[1],aResult ) )

flagPredicat = TRUE;

//

if( !flagNot ) // имя Не начинается с '!' или '~'

if( !strcmp( aPredicat,aResult ) )

flagPredicat = TRUE;

}

//

//--- теперь определяем значение Result на true или false и окончательно -------

//--- (с учётом статических true/false) устанавливаем flagPredicatTrue ---------

//

flagPredicatTRUE = FALSE; // начальная установка

if( flagPredicat ) // переменная-предиктор определена, но значение ещё неизвестно

if( ( flagNot && !Result ) || // имя начинается с '!' или '~' и Result==FALSE

( !flagNot && Result ) ) // имя Не начинается с '!' или '~' и Result==TRUE

flagPredicatTRUE = TRUE;

//

//--- отдельно обрабатываем статический true или false -------------------------

if( !strcmp( aPredicat, trueLowerCase ) ) // если true...

flagPredicatTRUE = TRUE;

if( !strcmp( aPredicat,falseLowerCase ) ) // если false...

flagPredicatTRUE = FALSE;

//

////////////////////////////////////////////////////////////////////////////////

//

if(Mem\_Instruction[i].fExec || // если инструкция ВЫПОЛНЯЕТСЯ "или"

Mem\_Instruction[i].fExecOut || // уже ВЫПОЛНЕНА "или"

Mem\_Instruction[i].fAddBuffer) // уже ДОБАВЛЕНА В БУФЕР

continue;

//

////////////////////////////////////////////////////////////////////////////////

//

switch( Get\_CountOperandsByInstruction( Set ) ) // !!! число входных операндов инструкции Set

{

case 1: // ----- один операнд + (возможно) предикат ------------------------

// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд) ...

if( !s\_isPredicat && !isPredicat )

{

if( !strcmp(Mem\_Instruction[i].aOp1, aResult) ) // 1-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp1 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го операнда

snprintf(tmp,sizeof(tmp), " %d(\*|1)", i); strcat(str, tmp); // флаг ГОТОВ у инструкции

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

//----- если это НЕ ПРЕДИКАТНАЯ инструкция, надо ещё проверить истинность поля ПРЕДИКАТА

if( !flagPredicat &&

flagPredicatTRUE ) // значение флага ПРЕДИКАТА

{

// Mem\_Instruction[i].fPredicat = TRUE; // установили флаг ПРЕДИКТОР\_ГОТОВ

Mem\_Instruction[i].fPredicatTRUE = TRUE; // установили флаг ПРЕДИКТОР\_ИСТИНЕН

snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i); strcat(str, tmp); // флаг TRUE у предиката

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( Mem\_Instruction[i].fOp1 && // первый операнд ГОТОВ...

flagPredicatTRUE ) // ... и флаг предикатора есть TRUE

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

} // конец if( !s\_isPredicat && !isPredicat )

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (1 операнд) ...

if( !s\_isPredicat && isPredicat )

{

if( !strcmp(Mem\_Instruction[i].aOp1, aResult) ) // 1-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp1 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го операнда

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( Mem\_Instruction[i].fOp1 ) // первый операнд предиката ГОТОВ... у предиката нет предиката

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

} // конец if( !s\_isPredicat && isPredicat )

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд) ...

if( s\_isPredicat && !isPredicat )

{

if( flagPredicatTRUE )

Mem\_Instruction[i].fPredicatTRUE = TRUE; // установим флаг предиката

//

if( Mem\_Instruction[i].fOp1 && // первый операнд ГОТОВ...

Mem\_Instruction[i].fPredicatTRUE ) // ... и флаг-предикат есть TRUE

{

snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i); strcat(str, tmp); // флаг TRUE у предиката

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

}

//

} // конец if( s\_isPredicat && !isPredicat )

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (1 операнд) ...

if( !s\_isPredicat && !isPredicat )

{

// if( !strcmp(Mem\_Instruction[i].aOp1,aResult) ) // 1-й операнд ГОТОВ

// {

// Mem\_Instruction[i].fOp1 = TRUE;

// snprintf(tmp,sizeof(tmp), " %d(1|1)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го операнда

// mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

// }

//

//----- если эта инструкция-НЕ ПРЕДИКАТ, надо проверить флаг (и его значение) поля предиката

if( !flagPredicat && // это инструкция НЕ ПРЕДИКАТ

flagPredicatTRUE ) // значение флага ПРЕДИКАТА

{

Mem\_Instruction[i].fPredicatTRUE = TRUE; // установили флаг ПРЕДИКAТ\_ИСТИНЕН

//if( i\_Set == 215 ) t\_printf( "\n43) %d/%s %d/%s Result=%f |aResult=\'%s\'|\n", i\_Set,Mem\_Instruction[i\_Set].Set, i,Mem\_Instruction[i].Set, Result, aResult );

snprintf(tmp,sizeof(tmp), " %d(PredTRUE|1)", i);

//if( i\_Set == 215 ) t\_printf( "\n44) %d/%s %d/%s Result=%f |aResult=\'%s\'|\n", i\_Set,Mem\_Instruction[i\_Set].Set, i,Mem\_Instruction[i].Set, Result, aResult );

strcat(str, tmp); // флаг TRUE предиката

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//if( i\_Set == 215 ) t\_printf( "\n45) %d/%s %d/%s Result=%f |aResult=\'%s\'|\n", i\_Set,Mem\_Instruction[i\_Set].Set, i,Mem\_Instruction[i].Set, Result, aResult );

//

if( Mem\_Instruction[i].fOp1 && // 1-й операнд ГОТОВ...

Mem\_Instruction[i].fPredicatTRUE ) // ... и флаг предиката есть TRUE

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

//

} // конец if( !s\_isPredicat && !isPredicat )

//

break; // конец обработки оператора с одним операндом + возможно, предиктор

////////////////////////////////////////////////////////////////////////////////

case 2: // ----- два операнда + (возможно) предикат ------------------------

// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (2 операнда) ...

if( !s\_isPredicat && !isPredicat )

{

if( !strcmp(Mem\_Instruction[i].aOp1,aResult) ) // 1-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp1 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(1|2)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го операнда

if( Mem\_Instruction[i].fOp1 && Mem\_Instruction[i].fOp2 )

{ snprintf(tmp,sizeof(tmp), " %d(\*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у инструкции

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( !strcmp(Mem\_Instruction[i].aOp2,aResult) ) // 2-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp2 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(2|2)", i); strcat(str, tmp); // флаг ГОТОВ у 2-го операнда

if( Mem\_Instruction[i].fOp1 && Mem\_Instruction[i].fOp2 )

{ snprintf(tmp,sizeof(tmp), " %d(\*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у инструкции

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

//----- если эта инструкция-НЕ ПРЕДИКАТ, надо проверить флаг (и его значение) поля предиката

if( !flagPredicat && // это инструкция НЕ ПРЕДИКАТ

flagPredicatTRUE ) // значение флага ПРЕДИКАТА

{

Mem\_Instruction[i].fPredicatTRUE = TRUE; // установили флаг ПРЕДИКAТ\_ИСТИНЕН

snprintf(tmp,sizeof(tmp), " %d(PredTRUE|2)", i); strcat(str, tmp); // флаг TRUE предиката

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( Mem\_Instruction[i].fOp1 && // первый операнд ГОТОВ...

Mem\_Instruction[i].fOp2 && // второй операнд ГОТОВ...

Mem\_Instruction[i].fPredicatTRUE ) // ... и флаг предиката есть TRUE

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

//

} // конец if( !s\_isPredicat && !isPredicat )

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - НЕ ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (2 операнда) ...

// if( !s\_isPredicat && isPredicat )

// break; // ... оператор - НЕ ПРЕДИКАТ не может изменять операнды у оператора - ПРЕДИКАТА

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - НЕ ПРЕДИКАТ (2 операнда) ...

if( s\_isPredicat && !isPredicat )

{

if( flagPredicatTRUE )

Mem\_Instruction[i].fPredicatTRUE = TRUE; // установим флаг предиката

//

if( Mem\_Instruction[i].fOp1 && // первый операнд ГОТОВ...

Mem\_Instruction[i].fOp2 && // второй операнд ГОТОВ...

Mem\_Instruction[i].fPredicatTRUE ) // ... и флаг предиката есть TRUE

{

snprintf(tmp,sizeof(tmp), " %d(PredTRUE|2)", i); strcat(str, tmp); // флаг TRUE у предиката

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

}

//

} // конец if( s\_isPredicat && !isPredicat )

//

// ----- ВЫПОЛНИВШИЙСЯ оператор - ПРЕДИКАТ && ЗАВИСИМЫЙ оператор - ПРЕДИКАТ (2 операнда) ...

if( s\_isPredicat && isPredicat )

{

if( !strcmp(Mem\_Instruction[i].aOp1,aResult) ) // 1-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp1 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(1|2)", i); strcat(str, tmp); // флаг ГОТОВ у 1-го операнда

if( Mem\_Instruction[i].fOp1 && Mem\_Instruction[i].fOp2 )

{ snprintf(tmp,sizeof(tmp), " %d(\*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у инструкции

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( !strcmp(Mem\_Instruction[i].aOp2,aResult) ) // 2-й операнд ГОТОВ

{

Mem\_Instruction[i].fOp2 = TRUE;

snprintf(tmp,sizeof(tmp), " %d(2|2)", i); strcat(str, tmp); // флаг ГОТОВ у 2-го операнда

if( Mem\_Instruction[i].fOp1 && Mem\_Instruction[i].fOp2 )

{ snprintf(tmp,sizeof(tmp), " %d(\*|2)", i); strcat(str, tmp); } // флаг ГОТОВ у инструкции

mI->Cells[6][i+1] = Vizu\_Flags(i); // визуализировали ФЛАГИ данной инструкции

}

//

if( Mem\_Instruction[i].fOp1 && // первый операнд ГОТОВ...

Mem\_Instruction[i].fOp2 ) // второй операнд ГОТОВ...

Add\_toBuffer( i ); // добавить ГКВ-команду в буфер команд для исполнения

} // конец if( s\_isPredicat && isPredicat )

//

break; // конец обработки оператора с двумя операндами + возможно, предикат

//

default: break;

//

} // конец switch по числу операндов у i-того оператора ---------------------

//

////////////////////////////////////////////////////////////////////////////////

} // конец цикла по пулу инструкций ..........................................

////////////////////////////////////////////////////////////////////////////////

//

////////////////////////////////////////////////////////////////////////////////

//

if(strlen(str)) // если в str что-то записывалось...

{

t\_printf( "-I- %s(){2}: по выполнению инструкции #%d/%d установлены флаги готовности операндов у инструкций: %s -I-",

\_\_FUNC\_\_, i\_Set, i\_Proc, str);

}

//

////////////////////////////////////////////////////////////////////////////////

// устанавливаем флаг однократного выполнения инструкции .......................

Mem\_Instruction[i\_Set].fExecOut = TRUE; // установили флаг ИНСТРУКЦИЯ\_ИСПОЛЬЗОВАНА

// снимаем флаг ИНСТРУКЦИЯ\_ВЫПОЛНЯЕТСЯ

Mem\_Instruction[i\_Set].fExec = FALSE; // сняли флаг ИНСТРУКЦИЯ\_ВЫПОЛНЯЕТСЯ

////////////////////////////////////////////////////////////////////////////////

mI->Cells[6][i\_Set+1] = Vizu\_Flags(i\_Set); // визуализировать...

////////////////////////////////////////////////////////////////////////////////

Draw\_ReadyOperands(); // выделение ячеек цветом

////////////////////////////////////////////////////////////////////////////////

//

Mem\_Proc[i\_Proc].Busy = FALSE; // АИУ номер i\_Proc теперь СВОБОДНО !!!

//

t\_printf( "-I- %s(){3}: АИУ номер %d освобождено (%s) после выполнения инструкции #%d -I-",

\_\_FUNC\_\_, i\_Proc, Get\_Time\_asLine(), i\_Set);

//

Free\_Proc ++ ; // число свободных АИУ увеличили на 1 ==========================

//

////////////////////////////////////////////////////////////////////////////////

sleep\_for\_vizu\_buffer // ждем-с для визуализации буфера

////////////////////////////////////////////////////////////////////////////////

AttemptExecMaxInstructions\_fromBuffer(); // пытаемся выполнить как можно больше ГКВ-инструкций из буфера

// для фактического выполнения инструкций из AttemptExecMaxInstructions\_fromBuffer()

// вызывается ExecuteInstructions\_ExceptET( i\_Set )

////////////////////////////////////////////////////////////////////////////////

sleep\_for\_vizu\_buffer // визуализируем буфер...

////////////////////////////////////////////////////////////////////////////////

//

Already\_Exec ++ ; // число уже испОлненных инструкций

//

Vizu\_Flow\_Exec(); // визуализация процента исполненных инструкций

//

} // --- конец FinalizeExceptSET\_DetectReadyForRun\_AddToBuffer\_AttemptExecMax\_Instructions