The control memory is initialized with the assembly code. The data memory starts from very next location where the control code finishes. Both program and data share the same memory due to mutually exclusive access to the memory and never overlap. The start address of the data memory is stored in the register as dms\_addr. Each instruction word is 16-bit long.

memory.wmf

**Store Immediate**

This instruction stores the immediate value provided in the instruction at the desired memory location as shown in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| Store\_imm | 42 | 01000010 | 2 | M[RZ] 🡨 Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Store immediate | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| pushzonstack | A 🡨 regZ |
| pushopdonstack | A 🡨 opdreg, B 🡨 A |
| wrdm | DM[B+dmbase] 🡨A |

**Store Idirect**

This instruction stores the content of Rx into memory location pointed to by Rz address.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| Store\_ind | C2 | 11000010 | 1 | M[RZ] 🡨 Rx |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Store\_indi | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| pushZ | A 🡨 regZ |
| pushX | A 🡨 regX, B 🡨 A |
| wrdm | DM[B+dmbase] 🡨A |
| Nop nxt | Fetch next bytecode |

**Store direct**

This instruction stores the content of Rx into memory location pointed to by direct address.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| Store\_dir | 82 | 10000010 | 2 | M[operand] 🡨 Rx |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Store immediate | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| pushOpd | A 🡨 opdreg |
| pushZ | A 🡨 regZ, B 🡨 A |
| wrdm | DM[B+dmbase] 🡨A |
| Nop nxt | Fetch next instruction |

**Load Immediate**

Load Rz with the content of immediate value.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| ldr\_imm | 40 | 01000000 | 2 | Rf[Z] 🡨 Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Ldr\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| pushOpd | A 🡨 opdreg |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**Load Indirect**

Load Rz with the content of memory location pointed to by Rx

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| ldr\_ind | C0 | 11000000 | 1 | Rf[Z] 🡨 Rx |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| ldr\_ind | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| pushX | A 🡨 regX |
| rdDM | A 🡨 rdDM[A] |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**Load Direct**

Load Rz with the content of memory location pointed to by opernd.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| ldr\_dir | 40 | 11000000 | 2 | Rf[Z] 🡨 M[Operand] |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| ldr\_dir | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg |
| rdDM | A 🡨 rdDM[A] |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**AND Immediate**

The contents of Rx and Operand are ANDed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| And\_imm | 48 | 01001000 | 2 | Rf[Z] 🡨 Rx & Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| And\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg |
| Push X | A 🡨 regX, B 🡨 A |
| And | A 🡨 A & B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**AND Indirect**

The contents of Rx and Rz are ANDed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| And\_ind | C8 | 11001000 | 1 | Rf[Z] 🡨 Rx & Rz |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| And\_ind | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| pushZ | A 🡨 regZ |
| And | A 🡨 A & B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**OR Immediate**

The contents of Rx and Operand are ORed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| or\_imm | 4C | 01001100 | 2 | Rf[Z] 🡨 Rx | Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| or\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg |
| Push X | A 🡨 regX, B 🡨 A |
| or | A 🡨 A or B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**OR indirect**

The contents of Rx and Rz are ORed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| And\_imm | CC | 11001100 | 1 | Rf[Z] 🡨 Rx or Rz |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| And\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| pushZ | A 🡨 regZ |
| And | A 🡨 A or B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**ADD Immediate**

The contents of Rx and Operand are ADDed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| addd\_imm | 78 | 01101000 | 2 | Rf[Z] 🡨 Rx + Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| add\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg |
| Push X | A 🡨 regX, B 🡨 A |
| add | A 🡨 A + B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**ADD Indirect**

The contents of Rx and Rz are ADDed and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| add\_ind | F8 | 11001000 | 1 | Rf[Z] 🡨 Rx + Rz |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| add\_ind | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| pushZ | A 🡨 regZ |
| add | A 🡨 A + B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**SUBV Immediate**

The contents of Rx and Operand are subtracted and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| subv\_imm | 43 | 01000011 | 2 | Rf[Z] 🡨 Rx - Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| subv\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg , B 🡨 A |
| sub | A 🡨 A - B |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**SUB Immediate**

The contents of Rx and Operand are subtracted and the result is stored in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| sub\_imm | 44 | 01000100 | 2 | Rf[Z] 🡨 Rx - Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| sub\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg , B 🡨 A |
| sub | A 🡨 A - B |
| Nop nxt | Fetch next instruction |

**JUMP Immediate**

Jump to address location unconditionally

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| jmp\_imm | 58 | 01011000 | 2 | PC 🡨 Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Jmp\_imm | Nop opd | Opdreg 🡨 CM[n+1] |
| Pushopd | A 🡨 opdreg , B 🡨 A |
| jbr | JPC 🡨A |
| pop | A 🡨 B |
| Nop nxt | Fetch next instruction |

**JUMP Indirect**

Jump to address location unconditionally

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| jmp\_imm | D8 | 11001000 | 1 | PC 🡨 Operand |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| Jmp\_imm | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX, B 🡨 A |
| jbr | JPC 🡨A |
| pop | A 🡨 B |
| Nop nxt | Fetch next instruction |

**Chkend**

Compare 4 bit memory blocks in Rx with Rz(3-0) and store the largest value back in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| chkend | EC | 11101100 | 1 | Rz <-MAX{Rx[15:12], Rx[11:8],Rx[7:4], Rx[3:0], Rz[3:0] } |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| chkend | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| findmax | regMax 🡨 max |
| pushmax | A 🡨regMax |
| wrrf | Rf[z] 🡨 A |
| Nop nxt | Fetch next instruction |

**Switch**

Compare 4 bit memory blocks in Rx with Rz(3-0) and store the largest value back in Rz

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Instruction | Opcode | Binary | Instr. words | Register Transfer |
| switch | E0 | 11100000 | 1 | Rz 🡨 M[Rx]  Rz 🡨 Rz + Rx + 1  PC 🡨 M[Rz] |

The equivalent instruction micro-instruction sequence in the TGALS.

|  |  |  |
| --- | --- | --- |
| switch | rdrf | regX 🡨rf[x], regZ 🡨 rf[z] |
| Push X | A 🡨 regX |
| rdDM | A 🡨 rdDM[A] |
| wrrf | Rf[z] 🡨 A |
| pushZ | A 🡨 regZ |
| pushX | A 🡨 regX, B 🡨 A |
| add | A 🡨 A + B |
| add | A 🡨 A + 1 |
| wrrf | Rf[z] 🡨 A |
| PushZ | A 🡨 regZ |
| jbr | PC 🡨 A |
| pop | A 🡨 B |
| Nop nxt | Fetch next instruction |