**ARM Instruction Set Architecture**

**Introduction**

ARM ISA is a set of instruction developed from the MIPS ISA, it is designed to be flexible and low powered embedded system. ARM processors can run software written in many high-level programming languages like C, C++, Java etc. It can also run low level languages like assembly.

**Instruction Classes**

ARM has similar instruction classes when compared to MIPS as they were developed by the same company. There is total 6 instructions classes in ARM ISA which are Data Processing Instruction, Data Transfer Instruction, Control Flow Instructions, Branch Instruction, Coprocessor Instruction and Miscellaneous Instructions but some of these instructions are a variant of a main instruction.

**Data Processing Instruction**

Data Processing Instruction is used to perform arithmetic and logic operations on operands such as ADD, SUB, AND, ORR, EOR and etc. The syntax is quite like MIPS.

Table

Description automatically generated

Figure : arithmetic and logic instructions of the ARM Assembly.

In ARM we have more instructions compare to MIPS for example we have CMP, CNP, ADC and etc. for example in ARM we have RSB (Reverse sub) which performs a subtraction operation by subtracting the value of the source operand from the value of the destination operand and storing the result in the destination register.

In Logical instruction, there are many new commands from MIPS like EOR (Exclusive OR) which is a logical operation used to perform XOR operation between two operands. EOR perform bitwise XOR operation between the value of the source operand and the value of the destination operand then it will store the result in the destination register, it is useful in situation where we need to manipulate individual bits of a register or memory location.

BIC (Bit Clear) is also a new operand that is new in ARM ISA and what it does it to clear one or more bits in a register or memory location. BIC instruction performs bitwise AND operation between the value of the source operand and the complement of the bit mast then it will store the result in the destination register.

In Register move instruction we have operands like LSL(Logical Shift Left) which take move destination register and source register or input value. The source register or input value specifies the bits to shift the value in the destination register to the left.

Conditional Instructions have operands such as CMP (Compare), TST (Test) for example CMP instruction performs a subtraction operation between two operands which in unlike the SUB instruction because it doesn’t store the result of subtraction but it will update the status to CPSR based on the result. It works by taking destination register and a source register or an immediate value and it compares the source register or immediate value to subtracted from the value in the destination register but the result is discarded.

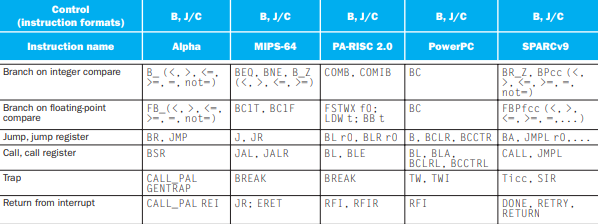
**Data Transfer Instruction**

Similar to MIPS assembly we have operands like LW and SW however we have LDR (Load Register) which is a data transfer instruction that loads a value from a memory location into a register. LDR operand will load the value from the memory location given by the memory address into the destination register which is similar to LW.

In ARM we can use STR (Store Register) which is used to store contents of a register into memory. The STR operand will store variables and data structure into a memory which is similar to SW.

Another operand that is useful in ARM is LDM (Load Multiple Registers) and STM (Store Multiple Registers) is useful when storing or loading multiple for example LDM will loads the values from the consecutive memory locations starting at the address given by the base register into the register listed in register list, the size of the values loaded depends on the type of LDM instruction used and STM works in a similar way.

**Control Flow Instruction**

****

**Table

Description automatically generated**

Figure The table of basic Control Flow operands

There are more operands for control flow instruction like the one we have from MIPS such as BEQ, BNE, BLT and etc.

Unconditional Control flows instruction like CALL is useful because it takes one operand which is the address of the function to be called which is similar to BNE but this one is simpler in my opinion.

Conditional Control Flow Instructions like JZ (Jump if Zero), JNZ (Jump if Not Zero) and etc. are quite useful as well and it is quite straight forward because it is based on the results of a comparison or a flag status.

**Summary**

Inconclusion, the MIPS and ARM Assembly share many common features but ARM has more features and unique operands that allows us to create more advanced features however it will also increases the skill ceiling when learning this language. The party I like about ARM Assembly is how it simplify many features in MIPS for example when looping to call a function we can simply use CALL instead of BEQ which is quite confusing for beginner like me.