Usage of the AVR assembly with the lstListing package

Dr. Benjamin Viall

August 15, 2018

1 Introduction

As of 2018 the ECE263 class uses the Atmel/Microchip atMega328p for its asm programming. this uses a assembly language varient which is unsupported natively by the lstlisting package. given that the atmel studio enviroment intelegently detects register names and not all registers in a give processor are available in every processor in the atmel family, syntax highlighting will always be ad hoc in LaTeX. To simplify the usage of listings in lab reports the avrListing.tex file has been released to make atmega328p code listings look "good".

2 Usage

Code listings in avr asm require only that the 'avrListing.tex' file is in your project and you include the following line in your preamle (near your other usepackage directives)

`\include{avrListing.tex}`

now code can be included in two ways.

1. inline listings (suitable for short code snippets)
2. file include (appendices that have full file listings)

2.1 Any listing

Regardless of the listing type you choose all asm snippets will share common markup language. best practice sugests that snippets are shown as either figures or code listings. for the purposes of example, we will show all examples using the figure environment.

While it is not strictly needed to redeclare `\lstset` everytime you make a figure, it will make copy/paste operations easier in the future, especially if you are using multiple languages in a document. Figure 1 demonstrates how to begin adding a figure that will contain the AVR assembly snippet. Two keyword arguments are demonstrated. the first `language=[AVR]{Assembler}`, tells the listing package that you wish to use the keywords defined for the AVR variant of an Assembler language. The second , `style=avrasm` tells the package to use the avrasm style designed to mimic the syntax highlighting of of atmel studio.

```
\begin{figure}
\label{fig:macros}
\lstset{language=[AVR]{Assembler},style=avrasm}
...
\caption{a caption for this ASM listing...}
\end{figure}
```

Figure 1: Any figure that contains a asm listing.

```
\begin{figure}
\label{fig:macros}
\lstset{language=[AVR]{Assembler},style=avrasm}
\begin{lstlisting}[frame=none]

;STore Imm. byte - Store byte to sram
;  Usage: STI [Addr16],Rt,[k8]
.macro STI ;
LDI @1,@2
STS @0,@1
.endmacro
end{lstlisting}

\caption{macros...macros everywhere}
\end{figure}
```

Figure 2: tex source for inline code listings.

```
;STore Imm. byte - Store byte to sram
;  Usage: STI [Addr16],Rt,[k8]
.macro STI ;
  LDI @1,@2
  STS @0,@1
.endmacro
```

Figure 3: a simple macro

2.2 Inline Listings

now that we have covered the use of any type of listing we demonstrate the use of the lstlisting enviroment. the code shown in Figure 2 displays an avrAsm macro as a figure. the resulting code is shown in Figure 3. The colors are a result of the style keyword. changing the contents of the style inside of avrListing.tex is possible but not recommended. if you wish to define your own style begin by copying the style definition into your tex document and giving it a unique name, then edit to your desired color scheme.

It is also still possible to use other languages with the avrListing.tex file. In Figure 4 and Figure 5 the differences between c and asm listings are clear. only the language keyword is changed. the usage of the frame keyword is used as a stylistic choice and is not needed.

```
#include <avr/io.h>
void main()
{
        while (1)
        {
                PIND=_BV(5);
        }
}
```

Figure 4: other languages still work

```
\begin{figure}
\label{fig:CListing}
\begin{lstlisting}[language=c,frame=single]
 #include <avr/io.h>
 void main()
 {
 while (1)
 {
  PIND=_BV(5);
 }
}
end{lstlisting}
\caption{other languages still work}
\end{figure}
```

Figure 5: tex code for a C style Listing.

```
; your name(s)
; group #
; lab number
; date
; any collaborators

 .nolist
 .include "m328pdef.inc"
 .list

 .listmac

;Load Word Immediate - Load immediate Word into X,Z,or Z

.macro ldi16 ; [X,Y,Z],k16
  LDI @0H,high(@1)
  LDI @0L,low(@1)
.endmacro

;Load Word - Load Word from SRAM
;  Usage: LDSW [RegH], [RegL], [Addr16]
.macro LDSW
    LDS @0, @2
    LDS @1, @2+1
.endmacro

;STore Imm. byte - Store byte to sram
;  Usage: STI [Addr16],Rt,[k8]
.macro STI ;
  LDI @1,@2
  STS @0,@1
.endmacro

;STore Imm. Word - Load Word to sram
;  Usage: STIW [Addr16],Rt,[k16]
.macro STIW
  STI @0,@1,HIGH(@2)
  STI @0+1,@1,LOW(@2)
.endmacro




;***********************************************************
; data space variable section
.dseg
.org SRAM_START
 srcAddress:   .byte 2
 dstAddress:   .byte 2
 numBytes:     .byte 1
; ***************************************************************

;***********************************************************
; interrupt vector table
 .cseg
 .org 0x0000
 JMP main
;***********************************************************
```