16 Mayıs 2018 Çarşamba

Sistem Programlama Assembly

Genel Bilgiler

Mikroişlemci 8 Register'a sahip.
4 Bayt kullanıcı tarafından yazılıp okunabilir.
İlk 5 tanenin adı, R0,R1,R2,R3,R4
Son üç özel;

- SP Stack Pointer
- FP Frame Pointer
- PC Program Counter

Komutlar

```
ld mem -> %reg     Memorydeki değeri registera yükler

st %reg -> mem     Registerdaki değeri memory yükler
st %r0 -> i        Registerdaki değeri i değişkenine yüker
st %r0 -> [fp+4]   Bellekte 4 bit ilerisine yazar.

mov %reg -> %reg   Registerdan registera değerini kopyala
mov #val -> %reg   Registera sabit değer atar

add %reg1, %reg2 -> %reg3
                   Add reg1 & reg2 and put the sum in reg3.
sub %reg1, %reg2 -> %reg3   Subtract reg2 from reg1.
mul %reg1, %reg2 -> %reg3   Multiply reg1 & reg2.
idiv %reg1, %reg2 -> %reg3  Do integer division of reg2 into
reg1.
imod %reg1, %reg2 -> %reg3  Do reg1 mod reg2.

push %reg           This subtracts the value of stack pointer
push #val           by value contained in reg or the constant
defined in val.

pop %reg            This adds the value of %reg or #val
pop #val            to the stack pointer.

jsr a                Call the subroutine starting at
instruction a.

ret          Return from a subroutine.

.globl i           Allocate 4 bytes in the globals segment
                         for the variable i.
```

Örnek 1

```
     .globl i
     .globl j
main:

     mov #1  -> %r0
     mov #2  -> %r1
     add %r0,%r1  -> %r1
     st  %r1 -> j
     st  %r0 -> i
        ret
```

```
int i;
int j;
main()
{
  i = 1;
  j = 2;
  j = i + j;
}
```

Örnek 2

```
main:
     push #8                  / i ve j için alan ayırıyor
     mov #1   -> %r0
     st  %r0 -> [fp-4]        / i'ye 1 atıyor
     mov #2  -> %r0
     st  %r0 -> [fp]          / j'ye 2 atıyor
     ld  [fp-4]  -> %r0
     ld  [fp]    -> %r1
     add %r0,%r1 -> %r1       / i ve j'yi topluyor, r1e atıyor
     st  %r1 -> [fp]          / toplam j'ye yazılıyor
          ret

     main()
     {
     int i, j;
     i = 1;
     j = 2;
     j = i + j;
     }
```

Örnek 3

```
a:
     mov #1 -> %r0
     ret
main:
     push #4
     jsr a
     st %
```

```
int a()
{
  return 1;
}
main()
{
  int i;
  i = a();
}
```

Örnek 4

```
a:
     push #4
     ld [fp+12] -> %r0
     add %r0, %g1 -> %r0
     st %r0 -> [fp]
     ld [fp] -> %r0
          ret
main:
     push #4
     mov #5 -> %r0
     st %r0 -> [sp]--
     jsr a
     pop #4
     st %r0 -> [fp]
          ret
```

```
int a(int i)
{
  int j;

  j = i+1;
  return j;
}

main()
{
  int i;

  i = a(5);
}
```

Örnek 5

```
a:
     push #4
     ld [fp+8] -> %r0
     mov #2, %r1
     mul %r0, %r1 -> %r0
     st %r0 -> [fp]
     ld [fp] -> %r0
          ret
main:
     push #4
     mov #5 -> %r0
     st %r0 -> [sp]--
     jsr a
     pop #4
     st %r0 -> [fp]
          ret
```

```
int a(int i)
{
  int j;
  j = 2;
  j = i*j;
  return j;
}

main()
{
  int i;
  i = a(5);
}
```