**LAPORAN APLIKASI MIKROKONTROLER**

**PRAKTIKUM VII**

Nama :Farrel Raditya Eduardi (181344010)

Kelas : 3NK

Pembimbing : 1. Ferry Satria BSEE.,MT.

2. Rahmawati Hasanah, S.ST., M.T

Topik : Operasi ATMega 8535 dengan Multiple 7 Segment



**POLITEKNIK NEGERI BANDUNG**

**BANDUNG**

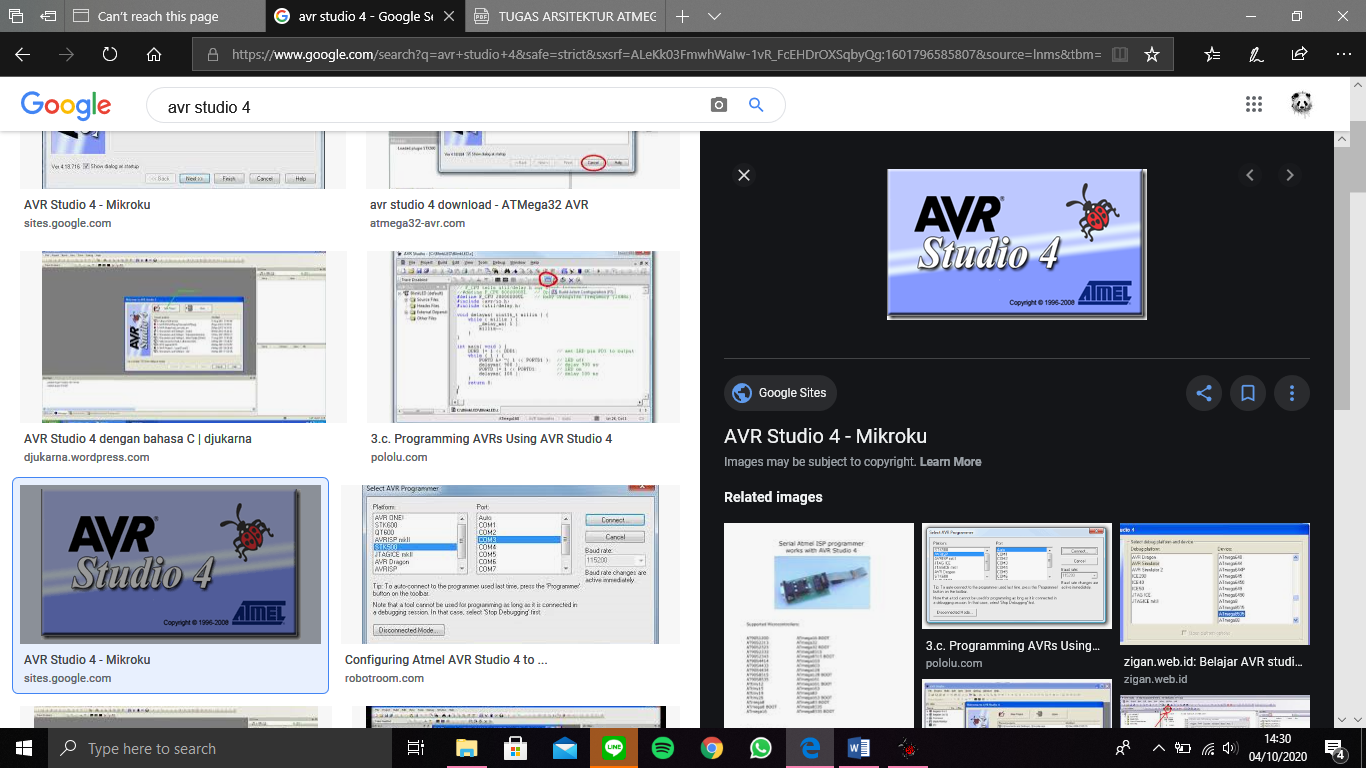
**2020**

Tanggal Praktikum : 12 November 2020

1. **Judul Praktikum**

Operasi ATMega 8535 dengan Multiple 7 Segment

1. **Landasan Teori**
2. AVR Studio 4



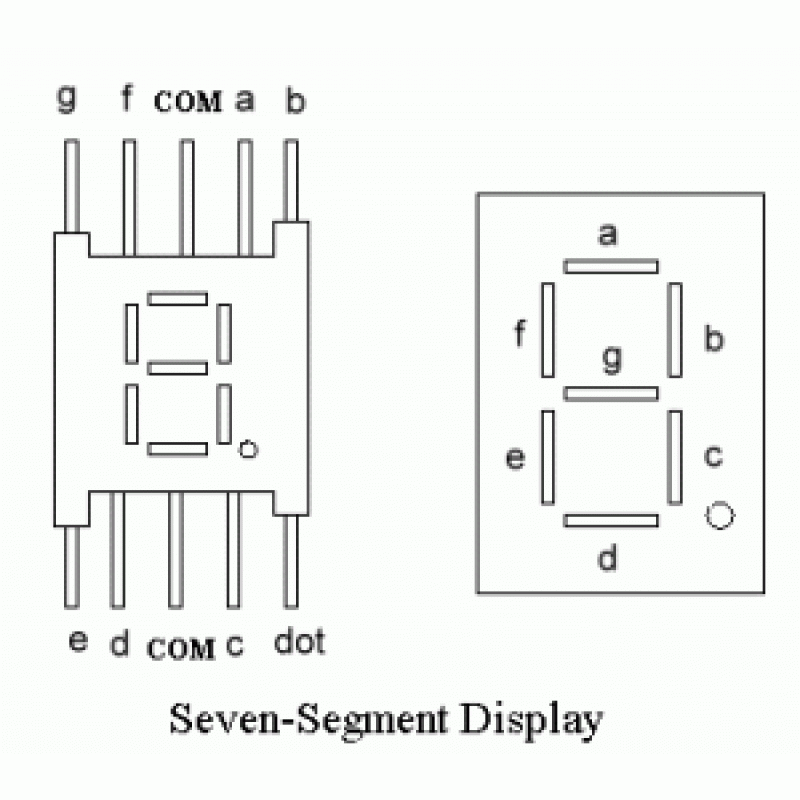
AVR studio adalah sebuah software Integrated Development Environment (IDE) yang dibuat oleh ATMEL untuk membuat aplikasi pemograman 8 bit pada mikrokontroller AVR. Pada dasarnya AVR studio 4 menggunakan bahasa pemograman Assembler. Bahasa Pemograman Assembler adalah bahasa tingkat rendah yang memiliki keunggulan tersendiri namun sulit untuk dipahami. Selain menggunakan bahasa pemograman Assembler, AVR Studio 4 juga dapat menggunakan bahasa C sebagai bahasa pemograman. Bahasa C adalah bahasa pemograman tingkat menengah yang lebih mudah untuk dipelajari bila dibandingkan dengan bahasa Assembler.

Untuk membuat program C dengan menggunakan AVR Studio 4, selain membutuhkan software AVR studio 4, kita juga membutuhkan include file yang terdapat di dalam program WinAVR. WinAVR adalah sebuah software opensource yang dibuat untuk memprogram mikrokontroller AVR yang bekerja pada sistem operasi windows.

Sebelum membuat suatu program di AVR Studio 4, harus dibuat terlebih dahulu “template” program yang mengatur stack pointer register dan pendefinisian ATMega 8535 itu sendiri. Stack Pointer Register dibagi 2 yaitu SPH (High) dan SPL (Low), adapun “template” program nya adalah

|  |  |
| --- | --- |
| .include “m8535def.inc” | //untuk mendifinisikan ATMega 8535 |
| .org 0x0000 | //origin, program akan dimulai dari alamat 0x0000 |
| RJMP main | //jump relatif ke main program |
| main: | //label main untuk main programnya |
| LDI r16, low (ramend) | //mengisi r16 dengan alamat akhir pada stack pointer low (spl) |
| OUT spl,r16 | //dari r16, dikeluarkan ke stack pointer low |
| LDI r16, high (ramend) | //mengisi r16 dengan alamat akhir pada stack pointer high (sph) |
| OUT sph,r16 | //dari r16, dikeluarkan ke stack pointer high |
| -MAIN PROGRAM- | -mulai melakukan program pada bagian ini- |
| Akhir : RJMP Akhir | //mengakhiri program |

1. Rangkaian 7 Segment

****Dekoder BCD ke 7 segment jenis TTL adalah rangkaian yang berfungsi untuk mengubah kode bilangan biner BCD (Binary Coded Decimal) menjadi data tampilan untuk penampil/display 7 segment yang bekerja pada tegangan TTL (+5 volt DC). Dekoder BCD ke 7 segmen yang digunakan adalah jenis TTL. Decoder BCD ke 7 segmen jenis TTL ada beberapa macam diantaranya keluarga IC TTL 7447 dan keluarga IC TTL 7448. Kedua IC TTL: tersebut memiliki fungsi yang sama namun peruntukannya berbeda IC 7447 digunakan untuk driver 7 segment common anoda sedangkan IC 7448 digunakan untuk driver dispaly 7 segment common cathode. IC dekoder BCD ke 7 segment sering juga dikenal sebagai driver display 7 segment karena selalu digunakan untuk memberikan driver sumber tegangan ke penampil 7 segment.

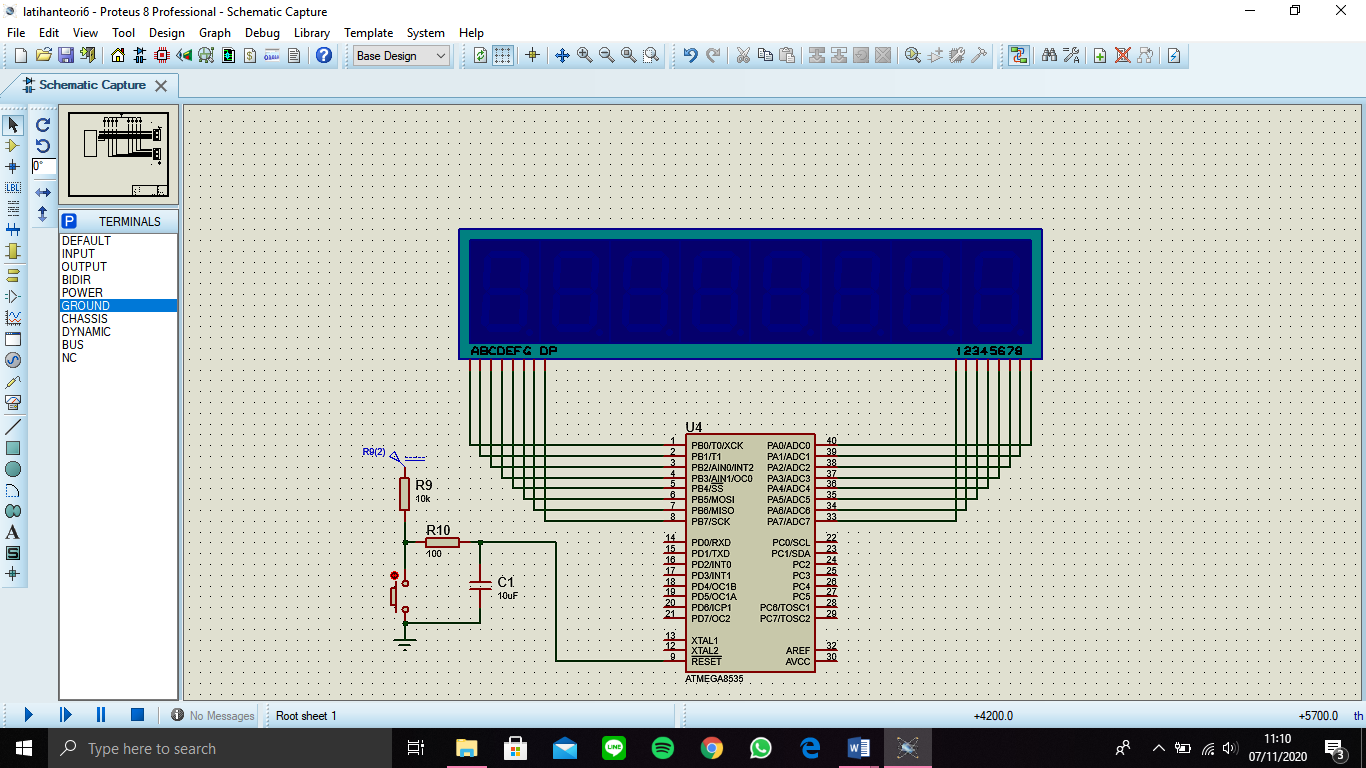
Konfigurasi Pin IC Dekoder BCD Ke 7 Segmen 7447 Dan 7448:

1. Jalur input data BCD, pin input ini terdiri dari 4 line input yang mewakili 4 bit data BCD dengan sebutan jalur input A, B, C dan D.
2. Jalur ouput 7 segmen, pin output ini berfungsi untuk mendistribusikan data pengkodean ke penampil 7 segmen. Pin output dekoder BCD ke 7 segmen ini ada 7 pin yang masing-masing diberi nama a, b, c, d, e, f dan g.
3. Jalur LT (Lamp Test) yang berfunsi untuk menyalakan semua led pada penampil 7 segmen, jalur LT akan aktif pad saat diberikan logika LOW pad jalut LT tersebut.
4. Jalur RBI (Riple Blanking Input) yang berfungsi untuk menahan sinyal input (disable input), jalur RBI akan aktif bila diberikan logika LOW.
5. Jalur RBO (Riple blanking Output) yang berfungsi untuk menahan data output ke penampil 7 segmen (disable output), jalur RBO ini akan aktif pada sat diberikan logika LOW.

Dalam aplikasi decoder, ketiga jalur kontorl (LT, RBI dan RBO) harus diberikan logika HIGH dengan tujuan data input BCD dapat masuk dan penampil 7 segmen dapat menerima data tampilan sesuai data BCD yang diberikan pada jalur input.

1. Rangkaian Multiple 7 Segment

Rangkaian dari Multiple 7 segment ini adalah sebagai berikut:



Pada teori dan praktikum ini digunakan yang anode. Input pada port B dihubungkan dengan pin A – DP dimana pin A menjadi LSB dan pin DP menjadi MSB. Untuk Output port A dihubungkan ke pin 1-8, dimana pin 8 menjadi LSB dan pin 1 menjadi MSB.

Urutan seven segment ini adalah bit 7 dari paling kiri sampai bit 0 di paling kanan. Gabungan dari Bit 7 sampai 0 ini disebut frame. Untuk membuat suatu rangkaian yang baik, dalam 1 detik harus terdapat minimal 30 frame dan lebih baik 60 frame. Perhitungannya apabila digunakan 60 frame (60fps),

1 detik = 60 frame

\*terdapat 8 seven segment sehingga terdapat 8 karakter yang terpampang

1 frame = 8 karakter

1 detik = 480 karakter (8 x 60)

Maka,

1 karakter = 1000/480 = 2ms

Maka tiap karakter ini harus muncul selama 2ms terus menerus. Waktu yang sangat cepat ini akan sulit terlihat mata dan akan terlihat terus menyala selama 1 detik.

Hal lain yang perlu diketahu adalah posisi digit 8 seven segment ini. Apabila diamati, 8 seven segment ini bisa dianalogikan menjadi 8 bit biner sebagai berikut:

1 1 1 1 1 1 1 1

Ini menandakan semua seven segment aktif, dan apabila 0 bisa dianggap semua seven segment mati, Maka apabila kita hanya ingin mengaktifkan seven segment bit ke 7 saja (paling kiri) maka kode biner nya menjadi:

1 0 0 0 0 0 0 0

Ini menandakan hanya bit 7 saja yang aktif. Dari urutan biner tersebut apabila dijadikan heksadesimal maka menjadi **8 0**. Dari prinsip ini, kode digit-digit seven segment nya menjadi:

Bit 7 = 80 (1(000 0000)

Bit 6 = 40 (0100 0000)

Bit 5 = 20 (0010 0000)

Bit 4 = 10 (0001 0000)

Bit 3 = 08 (0000 1000)

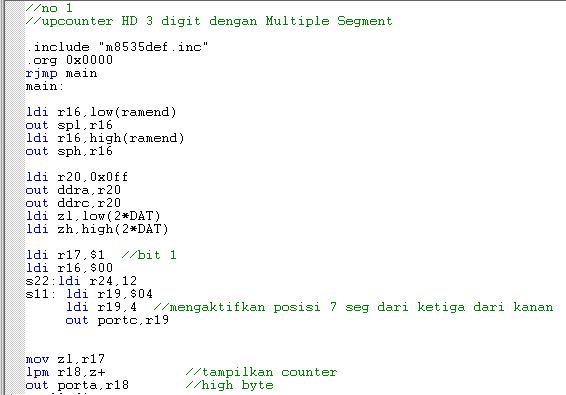
Bit 2 = 04 (0000 0100)

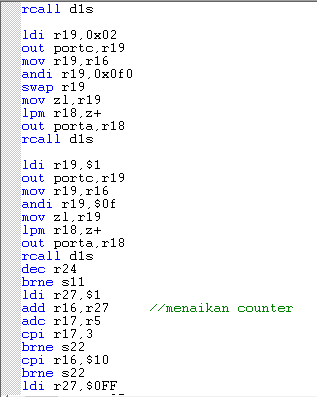
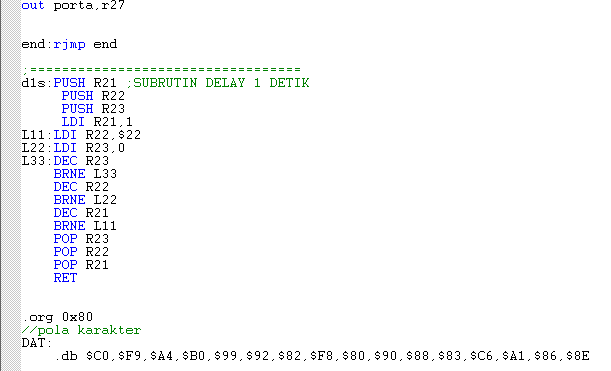
Bit 1 = 02 (0000 0010)

Bit 0 = 01 (0000 0001)

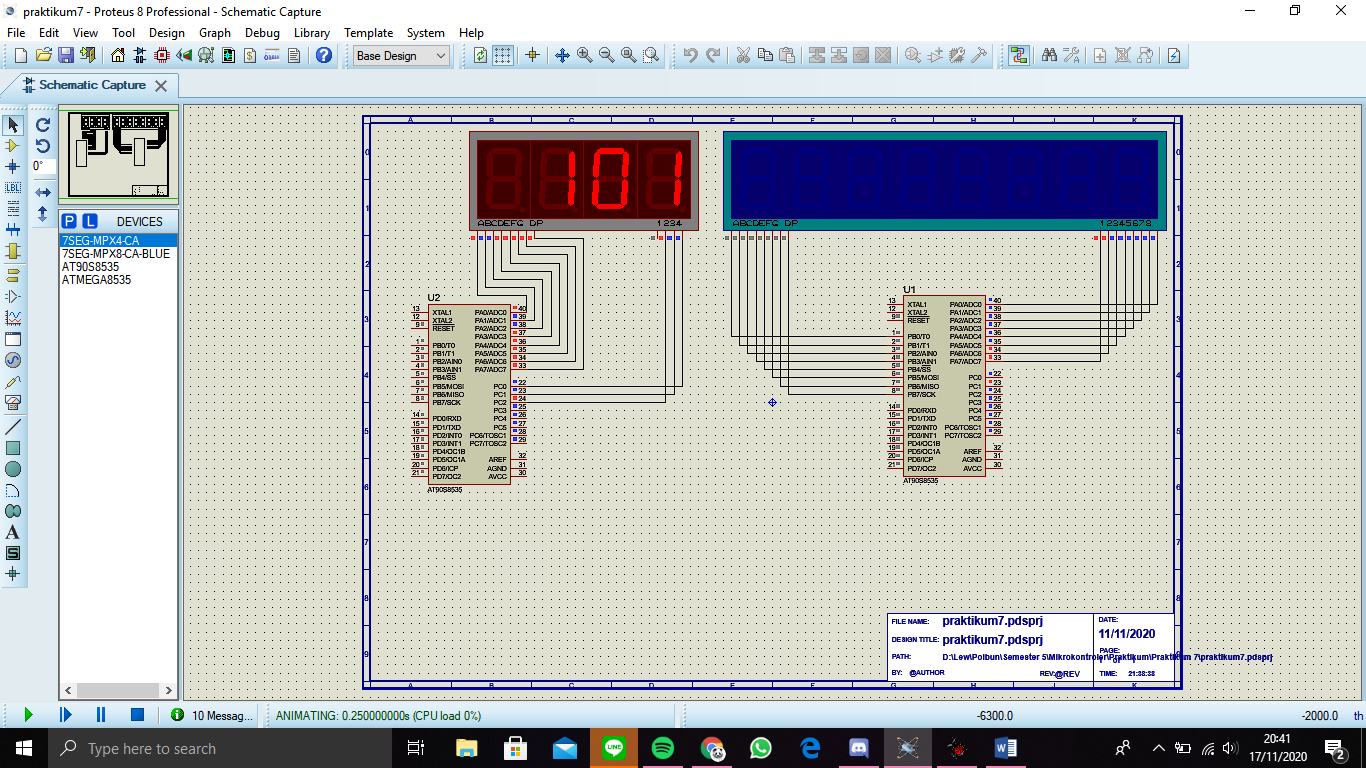
1. **Soal, Jawaban, dan Hasil Analisis**
2. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Tidak berulang.

Program:





Hasil:



Hasil Analisis:

Program ini adalah counter 3 digit heksadesimal dari 100 – 30F. Digunakan multiple 7 segment berjenis common anode. Bit paling rendah disimpan paling kanan dengan kode posisi digit yaitu $1 dan bit paling besar disimpan di paling kiri dengan kode posisi digit $4.

Pada Rangkaian, hubungkan port C ke pin 4,3,2 dan port A dihubungkan dengan pin A-G,Dp. Kedua port ini bertindak sebagai output maka itu diberi nilai FF agar dapat bertindak sebagai output. Digunakan program area DAT yang berisi kode posisi dan kode karakter heksadesimal (0-F).

Port C akan bertindak sebagai output kode posisi digit dan Port A sebagai output kode karakter. Dari soal, digit paling kiri langsung di outputkan “1”, digit kiri ini akan digunakan counter R17. Sementara itu, untuk digit tengah dan paling kanan digunakan register R16 sebagai counternya. Dimulai dengan memprogram digit paling kiri, maka dimasukan kode posisinya yaitu $4 dioutputkan ke port C. Untuk melakukan *load* data dari program area, digunakan register R18 untuk mengambil kode karakter “1” yang lalu di outputkan ke port A sehingga tercetak 1 pada 7 segment paling kiri.

Dilanjutkan untuk memprogram digit tengah dengan kode posisinya $2. Digunakan counter R16 yang berinisiasi 00, lalu di load program memory nya untuk mengambil kode karakter “0” di outputkan ke port C sehingga muncul pada digit tengah. Counter R16 ini digunakan juga untuk digit paling kanan dengan status output yang berbeda. Maka dari itu nilai di R16 nya ini harus diambil perenambelasannya saja. Caranya dengan memindahkannya ke R19 yang lalu di beri logic AND dengan 0xF0 sehingga terambil perenambelasannya, tetapi nilai ini masih berada di posisi perenambelasan, maka dari itu digunakan SWAP sehingga tertukar dengan low nibble nya, sehingga posisi nya menjadi di satuan kembali. Data yang ada di satuan ini akan menjadi alamat pointer z, lalu di load datanya (kode karakter) di R18 lalu dioutputkan ke digit tengah.

Dilanjutkan untuk memprogram digit 3 atau paling kanan, Digunakan kode posisi $1 selaku posisi digit paling kanan. Digunakan counter R16 sebelumnya yang lalu di proses pada R19 untuk diambil satuannya dengan AND $0F. Tidak perlu di swap karena sudah pada posisi satuan. Nilai tersebut dijadikan counter yang lalu dilakukan load data ke R18 yang di outputkan ke Port C sehingga tercetak di 7 segment paling kanan.

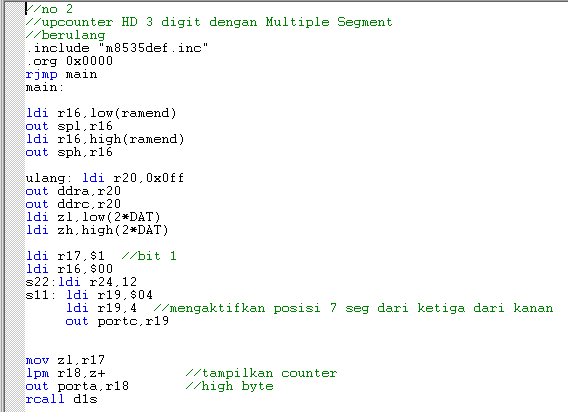
Dengan menaruh pola karakter pada alamat yang sama dengan karakter kita bisa dengan mudah menggunakan 1 register sebagai counter dan juga alamat sekaligus. Misalnya pertama kita ingin mengeluarkan 00, maka kita bisa isikan 00 ke suatu register kemudian satuan dijadikan alamat untuk mengambil data dan begitu juga dengan puluhan karena data tersebut akan ada pada alamat yang sama dengan datanya. Kemudian jika kita ingin mengeluarkan data 12 maka data tersebut bisa dijadikan alamat yang sudah mengandung pola karakter dari data tersbeut. Begitu lah yang dilakukan dengan R16 sebagai counter dan alamat sekaligus untuk digit satuan dan puluhan dengan cara yang sudah di jelaskan di paragraf sebelumnya.

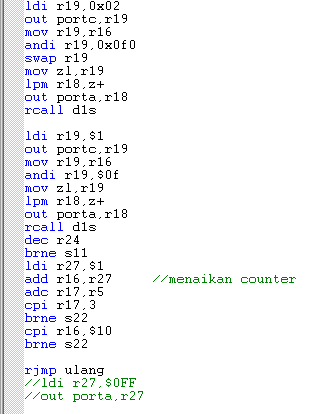
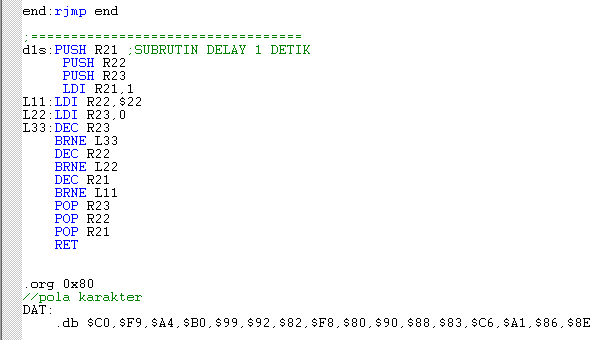
Agar nilai counter R16 dan R17 terus meningkat, nilai R16 ini ditambahkan 1 setiap selesai proses dan carry nya di tambahkan ke R17 selaku counter ratusan. Untuk menghentikan counter, dilakukan counter digit paling kiri dengan $3 atau R17 dengan $3. Apabila belum mencapai $3, maka digit tengah dan digit paling kanan terus melakukan counter dan prosesnya (diloncatkan kembali ke s22). Apabila sudah mencapai $3, maka pembanding selanjutnya adalah $10 untuk digit tengah dan akhirnya, yang dimana data terakhir nya menjadi 30F.

Apabila persyaratan ini sudah tercapai, masukan nilai FF ke r27 dan dioutputkan ke port A sehingga semua 7 segment menjadi mati, tandanya program sudah selesai.

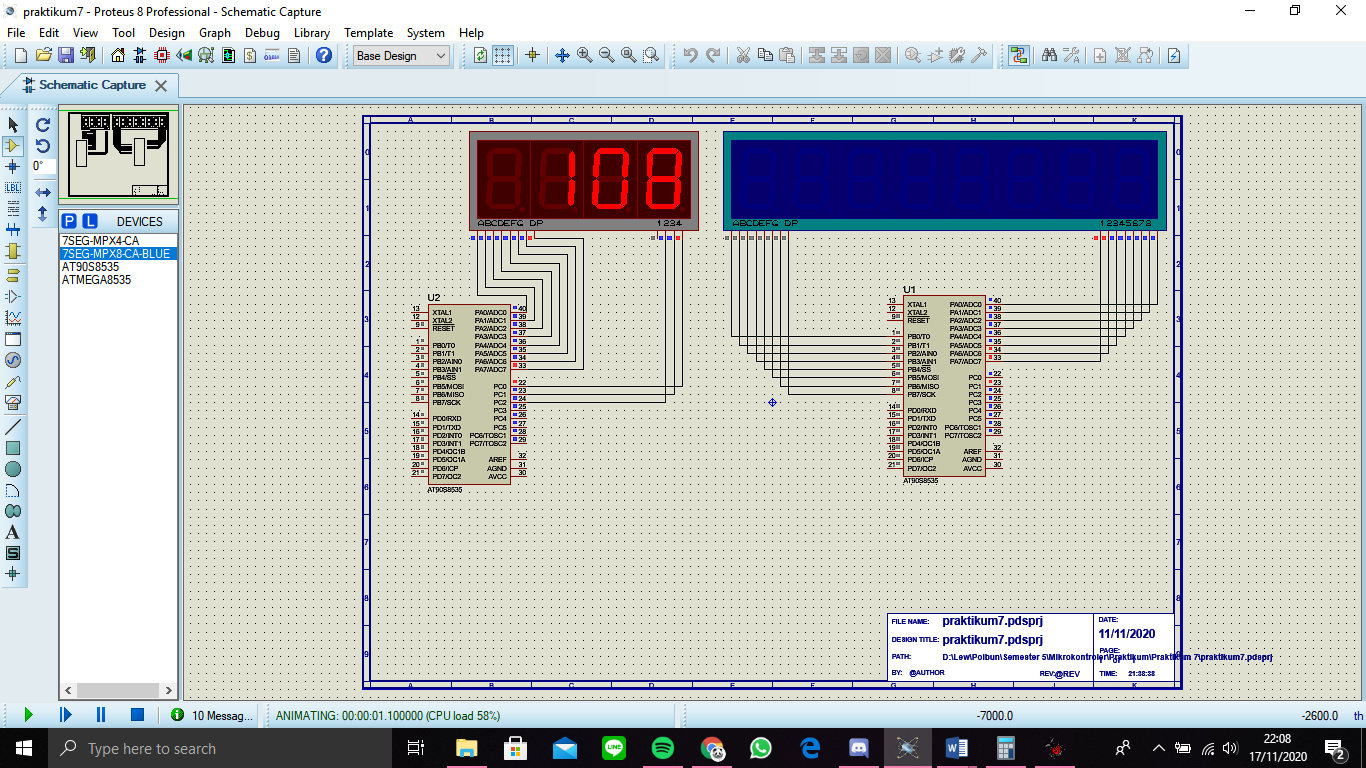
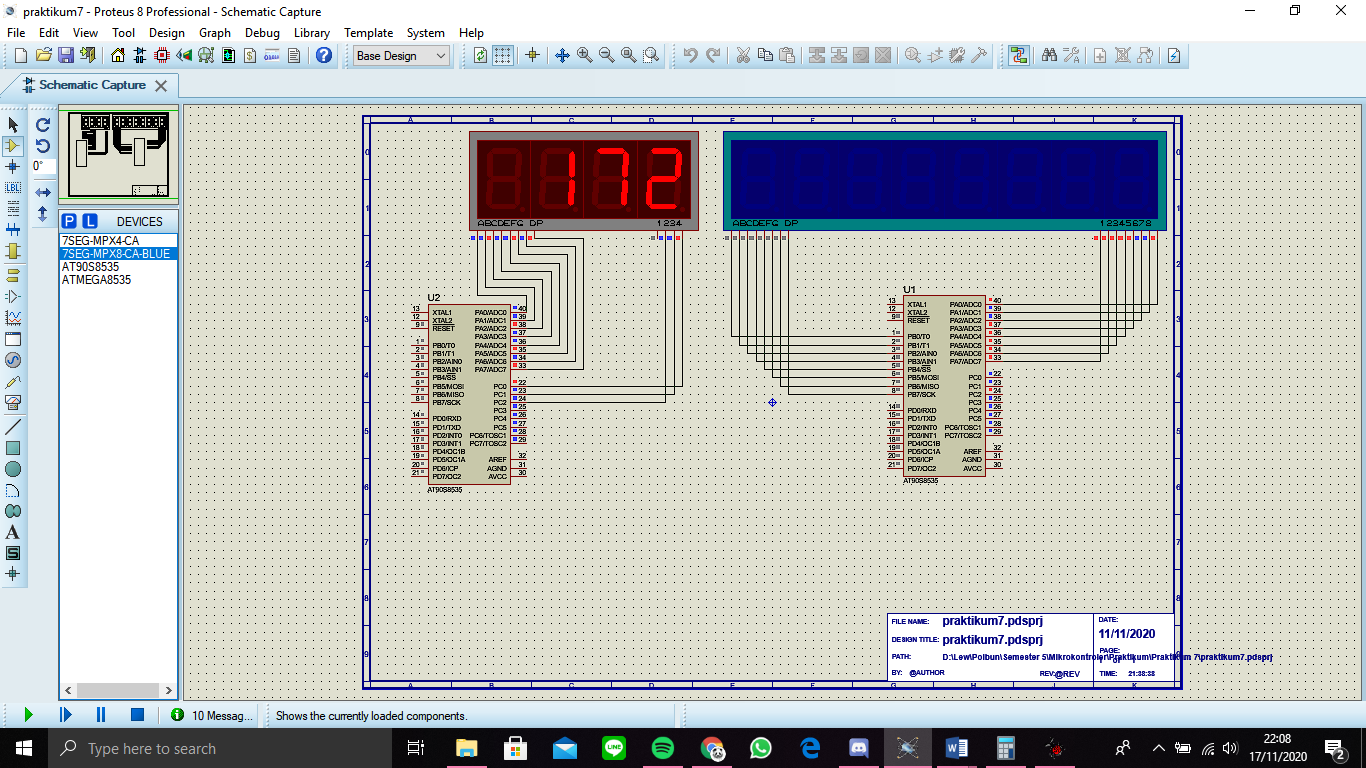
1. Buat program untuk Heksa-Desimal up-counter 3 digit (menghitung dari $100 sampai $30F) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.

Program:





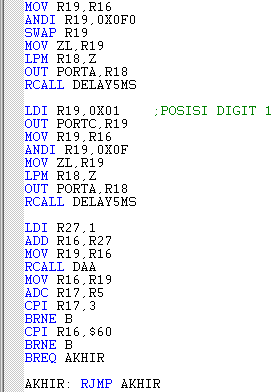
Hasil:

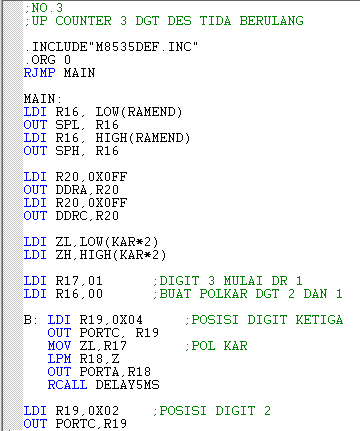
 

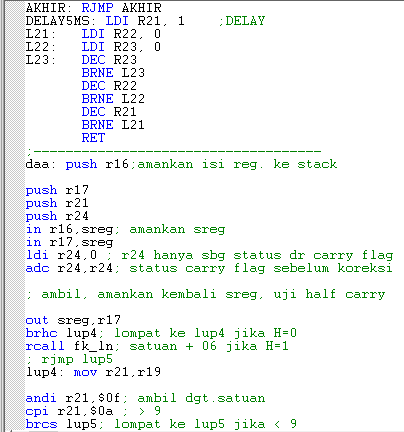
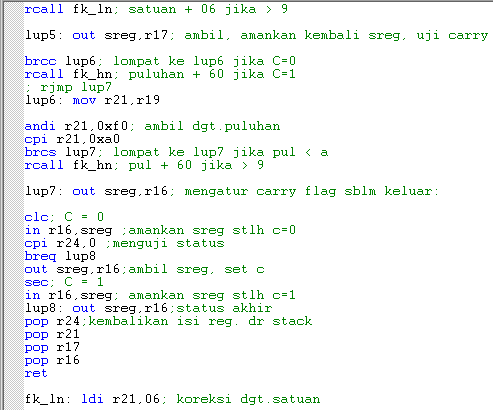
Hasil Analisis:

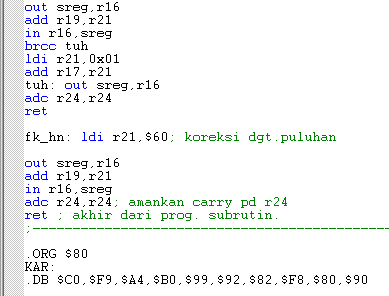
Program nomor 2 ini adalah ekstensi dari program no 1, dimana ditambahkan pengulangan yaitu dengan cara menambahkan label “ulang” diawal program (mulai dari mengatur port A dan C sebagai output) lalu pada akhir program ditambahkan rjmp ulang agar program terus berulang. Bagian pengisian data r27 dengan FF untuk mematikan 7 segment dihilangkan agar counter terus berulang tidak berhenti.

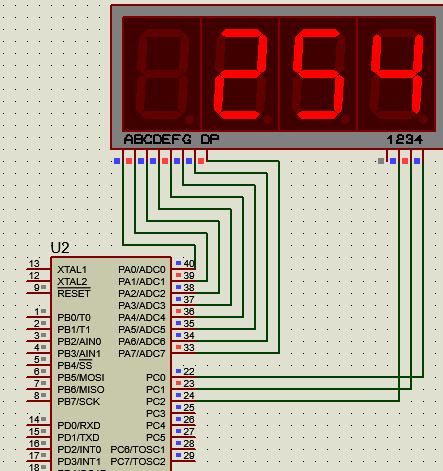
1. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Tidak berulang.

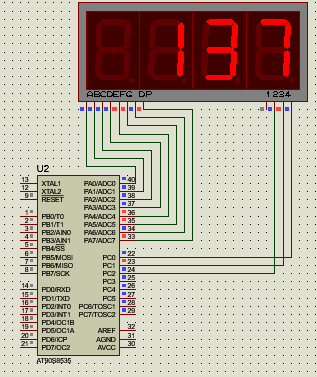
Program:







Hasil:

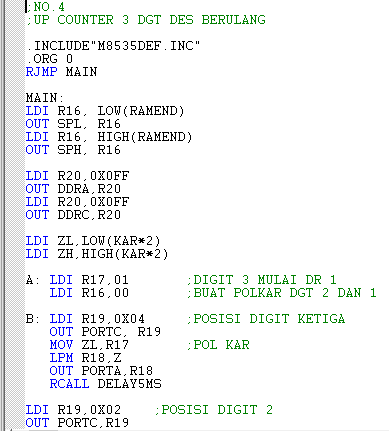
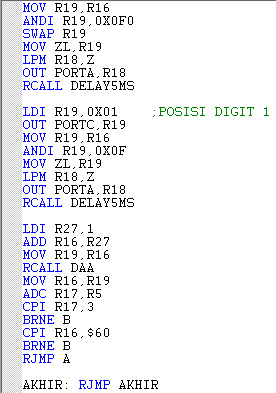


Hasil Analisis:

Program nomor 3 ini tidak jauh berbeda prosesnya dengan program nomor-nomor sebelumnya (terutama no 1). Perbedaan adalah, data yang di gunakan adalah desimal 100-359 sehingga dibutuhkan faktor koreksi pada counter R16 nya yaitu digunakan DAA agar digitnya tidak melebihi 9. Counter digit tengah dan digit kanan pun dibatasi sampai $60 sehingga data akhir nya yaitu 359. Pada program data pun, kode karakter yang dimasukan adalah pola karakter 0-9.

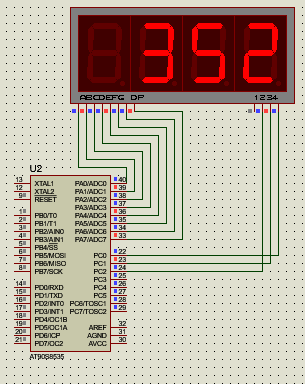
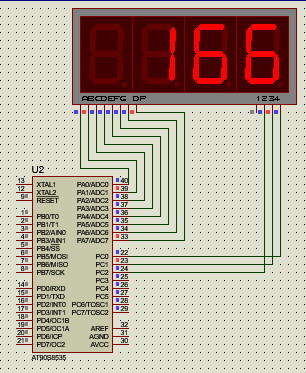
1. Buat program untuk Desimal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.

Program:



(Untuk Delay, subrutin DAA, dan program area sama seperti no 3)

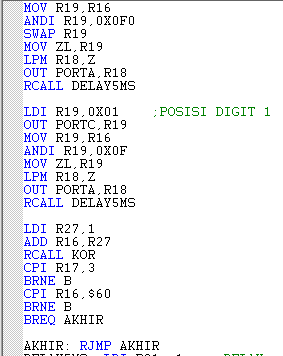
Hasil:

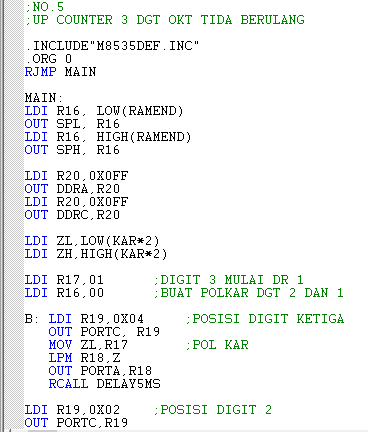


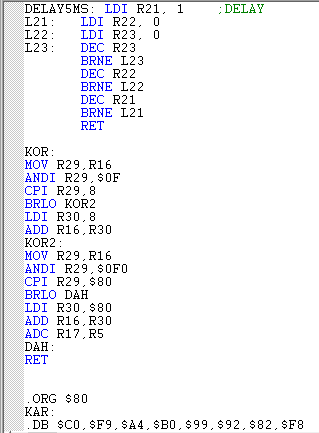
Hasil Analisis:

Program no 4 ini adalah pengembangan dari soal nomor 3. Perbedaan adalah program counter ini dilakukan berulang-ulang dengan cara membuat label A pada awal program, lalu saat program telah selesai mencapai 359, program akan loncat lagi otomatis ke awal dengan rjmp A sehingga akan terjadi infinite loop atau berulang terus menerus.

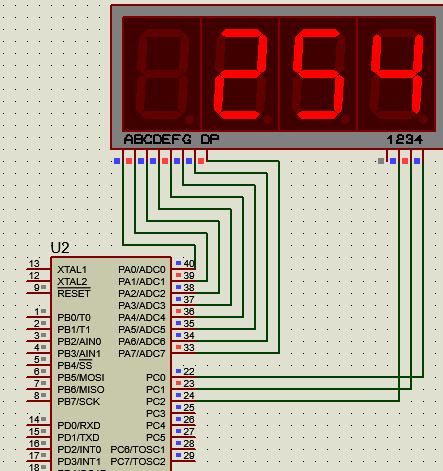
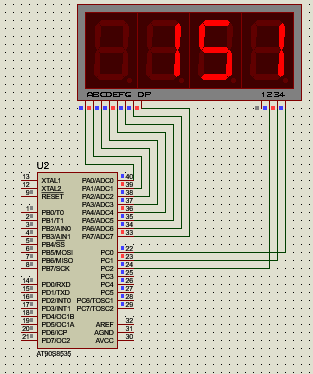
1. Buat program untuk Octal up-counter 3 digit (menghitung dari 100 sampai 357 yang outputnya berubah 1/5 detik sekali. Tidak berulang.

Program:





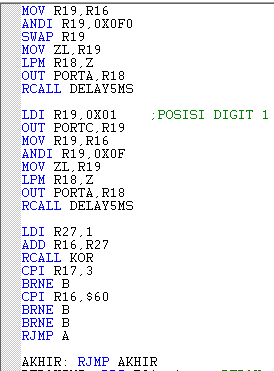
Hasil:

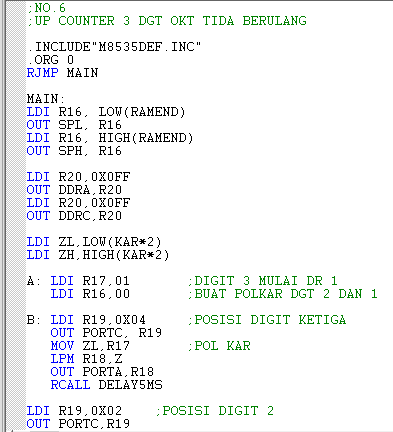


Hasil Analisis:

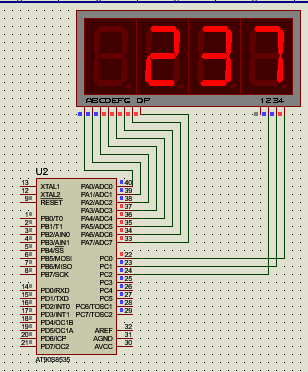
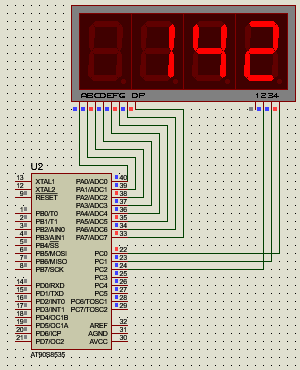
Program nomor 5 ini akan melakukan operasi up counter bilangan oktal dari 100-357. Prosesnya tidak berbeda jauh dengan program dengan soal sebelumnya. Perbedaan terletak pada digunakannya subrutin KOR yang akan mengoreksi bilangan sehingga tidak melebihi angka 7. Compaare hasil akhir pun dibatasi sampai $60 sehingga nilai akhirnya adalah 357. Program area pun hanya dimasukan kode karakter 0-7.

1. Buat program untuk octal up-counter 3 digit (menghitung dari 100 sampai 359) yang outputnya berubah 1/5 detik sekali. Berulang tanpa henti.

Program:



(Subrutin KOR, delay, dan program area sama seperti no 5)

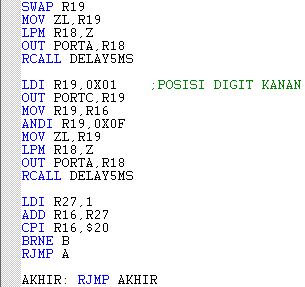
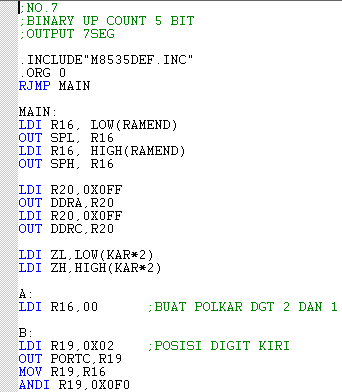
Hasil:

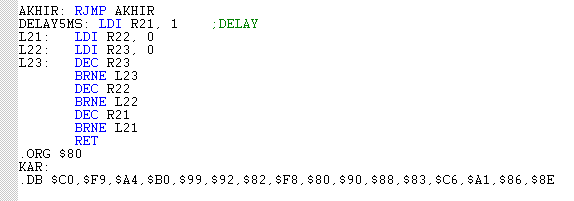
Hasil Analisis:

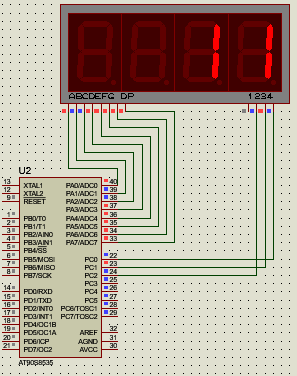
Progaram nomor 6 ini adalah pengembangan dari nomor 5. Program counter oktal ini akan dilakukan berulang-ulang tanpa henti. Cukup ditambahkan label baru (A) pada awal inisiasi program, lalu ditambahkan rjmp A pada akhir proses (setelah mencapai 357 sehingga kembali ke 100) sehingga terjadi pengulangan secara berulang terus menerus.

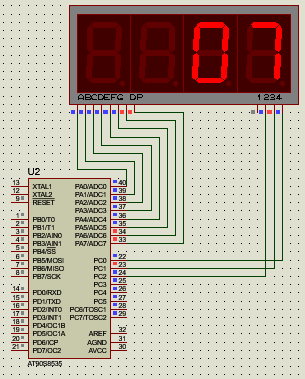
1. Buat program untuk Binary up-counter 5 bit yang outputnya berubah 1 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.

Program:



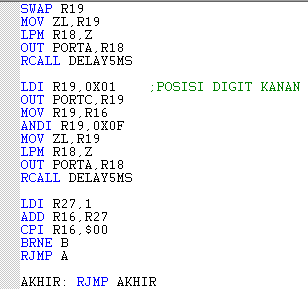
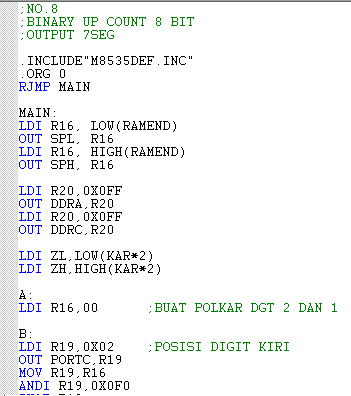


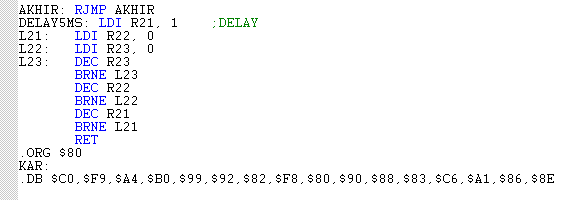
Hasil:

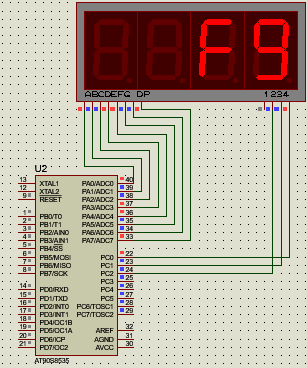


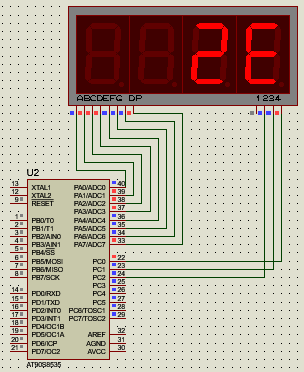
1. Buat program untuk Binary up-counter 8 bit yang outputnya berubah 1/4 detik sekali, menghitung berulang terusmenerus. Gunakan unit peraga 7 segment sebagai output.

Program:





Hasil:



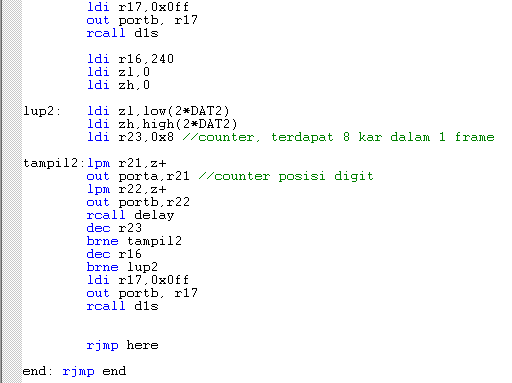
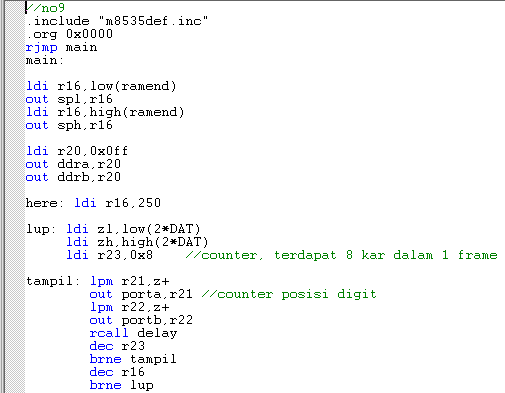
Hasil Analisis (7 & 8):

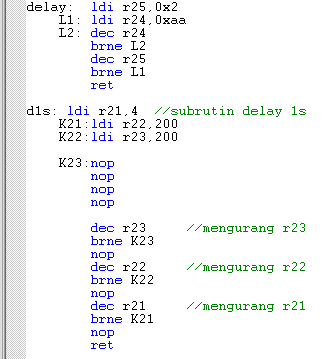
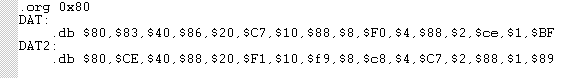
Program nomor 7 dan 8 ini adalah program up counter 5 bit (no 7) dari data 00 – 1F dan up counter 8 bit (no 8) dari 00 – FF. Operasi yang digunakan cukup sama dan mengacu pada nomor-nomor sebelumnya dengan beberapa perbedaan. Counter yang digunakan hanya 1 karena tidak adanya digit ketiga yang digunakan. Dilakukan program sama seperti sebelumnya yaitu pertama digit MSB dahulu ($2) lalu digit LSB paling kanan ($1), caranya cukup sama dengan mengatur digit tengah dan digit paling kanan seperti soal sebelumnya. Digunakan juga kembali kode karakter heksadesimal, sehingga kodenya dari 0 – F.

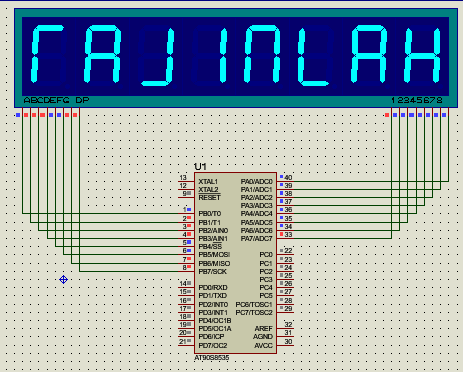
Perbedaan kedua nomor ini adalah batas counternya (R16). Pada nomor 7 karena data maksimalnya 1F, maka dilakukan compare hingga $20. Untuk nomor 8, karena datanya maksumumnya FF maka dicompare dengan $00.

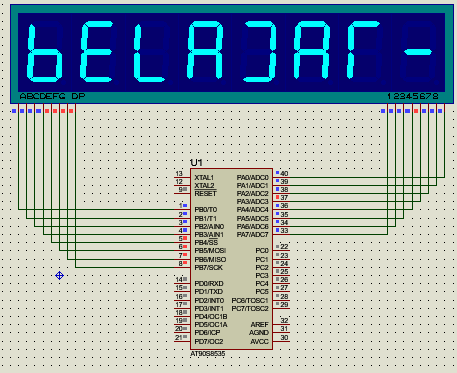
1. Buat program untuk menampilkan 2 kata pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata pertama: “rAJInLAH”. Kata ke dua: “bELAJAr-”.

Program:





Hasil:



Hasil Analisis:

Program nomor 9 ini sangat berbeda dengan program sebelumnya karena bukan sebuah program counter. Program ini menggunakan output Multiple 7 Segment 8 Digit Common Anode. Digunakan Port A dan Port B sebagai output, dimana Port A sebagai output kode posisi, dan port B sebagai output kode karakter.

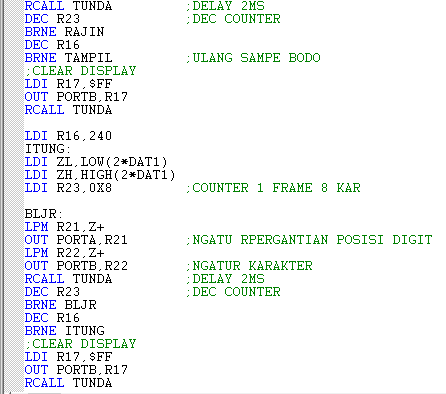
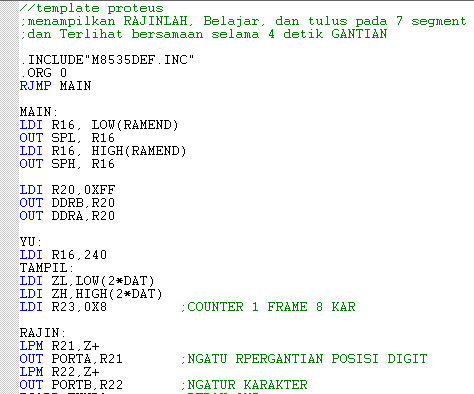
Pada program area diatur posisinya yaitu kode posisi dilanjut ke kode karakter lalu dilanjut kode posisi selanjutnya lalu kode karakter selanjutnya dan seterusnya hingga semua kode posisi sudah terpakai, Pada program, dimasukan terlebih dahulu kode posisi $80 sehingga urutan pengisiannya adalah dari kiri ke kanan. Pada program area, digunakan 2 label DAT dan DAT2 dimana DAT sebagai penyimpan kode posisi dan kode karakter “belajar”, dan DAT2 digunakan sebagai kode posisi dan kode karakter “rajinlah”.

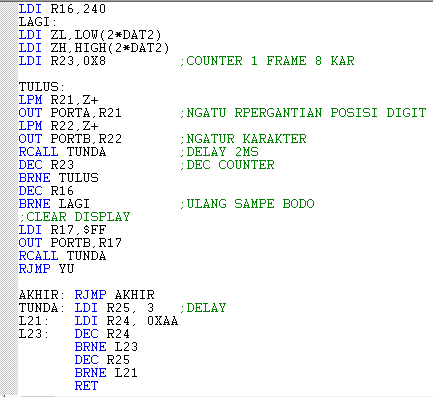
Program akan mencetak kata “belajar” dahulu. Diberikan inisiasi counter frame R16 sebanyak 240. Ini dimaksudkan agar kata “belajar” akan terus mencetak selama 4 detik sebelum ke proses selanjutnya. Untuk mengambil karakter “belajar” terlebih dahulu, maka lakukan load data dari program area DAT dahulu ke pointer z. Digunakan counter sebanyak 8 kali karena terdapat 8 posisi dan 8 karakter. Dilakukan pengoutputan kode posisi dahulu lewat load data program memory ke r21 lalu dioutputkan ke port A. Dilanjutkan dengan pengoutputan kode karakter lewat load data program memory ke r22 lalu dioutputkan ke port B. Proses ini akan berlangsung 8 kali sampai semua kode karakter dari “belajar” sudah tercetak. Ketika sudah mencapai angka 8 ini, program akan terus mengulang hingga counter R16 (240) terus berkurang hingga 0.

Setelah proses tersebut, program akan mereset seluruh 7 segment dengan memberi output port b dengan FF. Setelah clear, lakukan proses yang sama seperti sebelumnya hanya saja dengan kata “rajinlah”, maka dari itu digunakan label DAT2 untuk memanggil kode karakter “rajinlah”, selebihnya prosesnya sama. Pada semua kode karakter “rajinlah” sudah muncul dan counter frame nya sudah habis, loncatkan kembali ke awal program dengan **rjmp here** agar data muncul berulang-ulang.

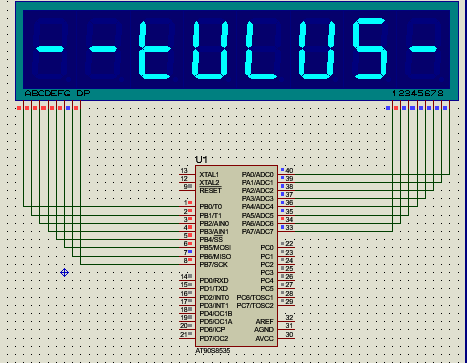
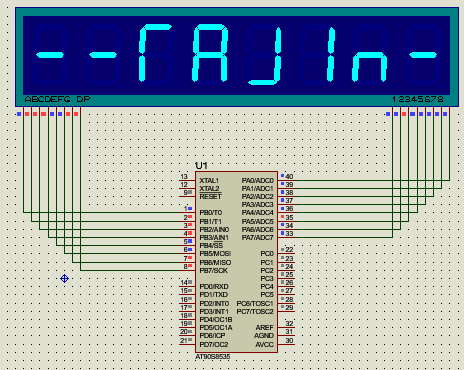
1. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik.

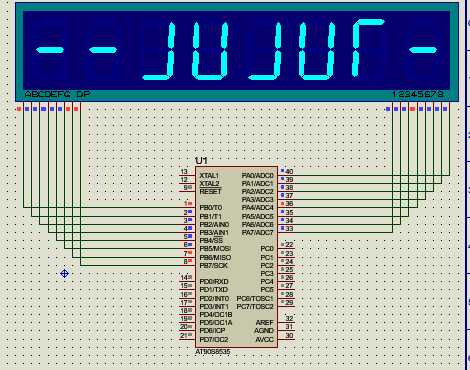
Program:





Hasil:



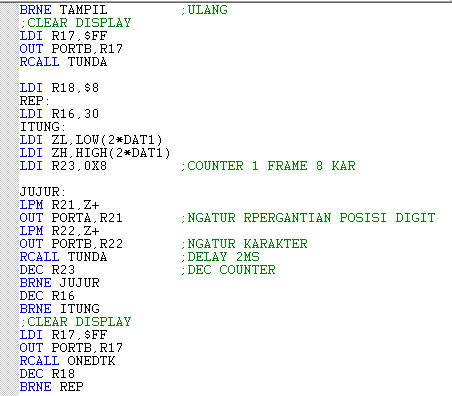
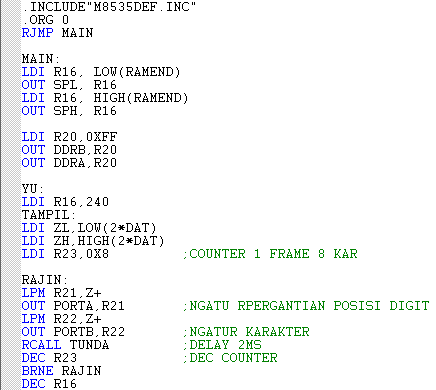


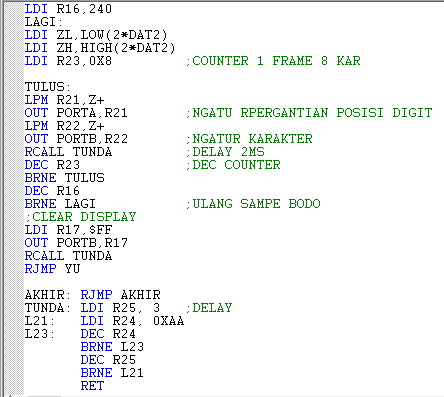
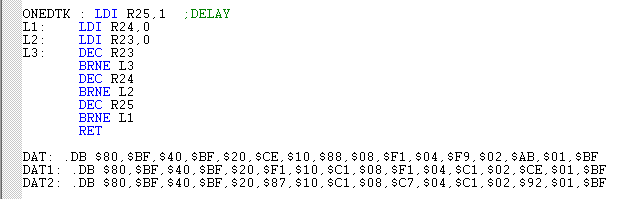
Hasil Analisis:

Program no 10 ini tidak jauh berbeda dengan program no 9. Terdapat 3 kata yaitu “—rajin-“, “—tulus-“, dan “—jujur-“ sehingga diperlukan 3 label pada program memory yairu DAT,DAT1, dan DAT2. Proses nya cukup sama yaitu dengan menggunakan counter frame r16 240 kali tiap katanya lalu dipanggila kode posisi dan kode karakternya.Karena terdapat 3 kata, maka proses ini menjadi 3 kali. Tidak lupa juga untuk selalu melakukan clean display dengan memberi nilai FF yang dioutputkan ke port b. Karena katanya berbeda dengan nomor sebelumnya, dibuat terlebih dahulu pola karakternya yang baru.

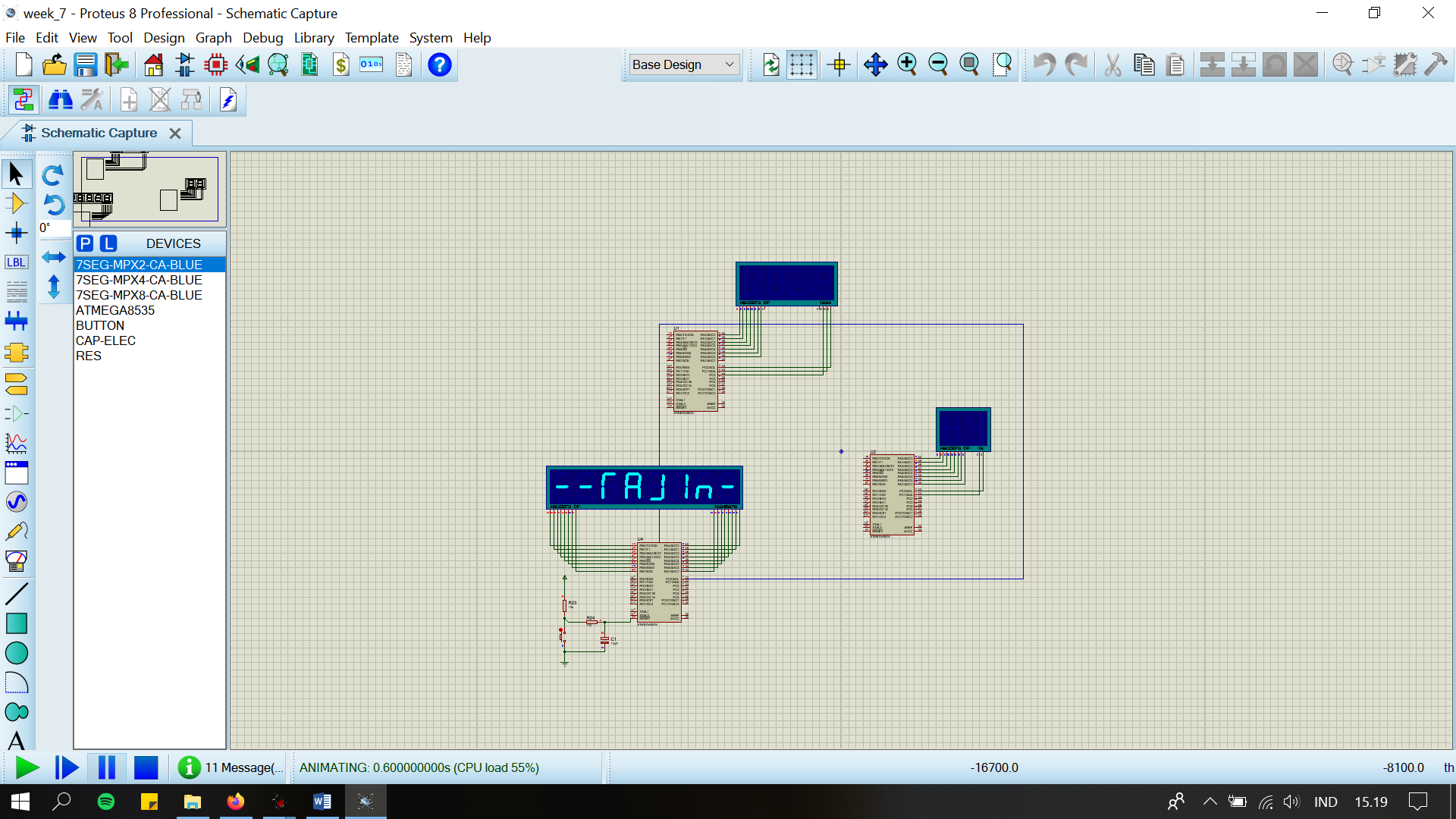
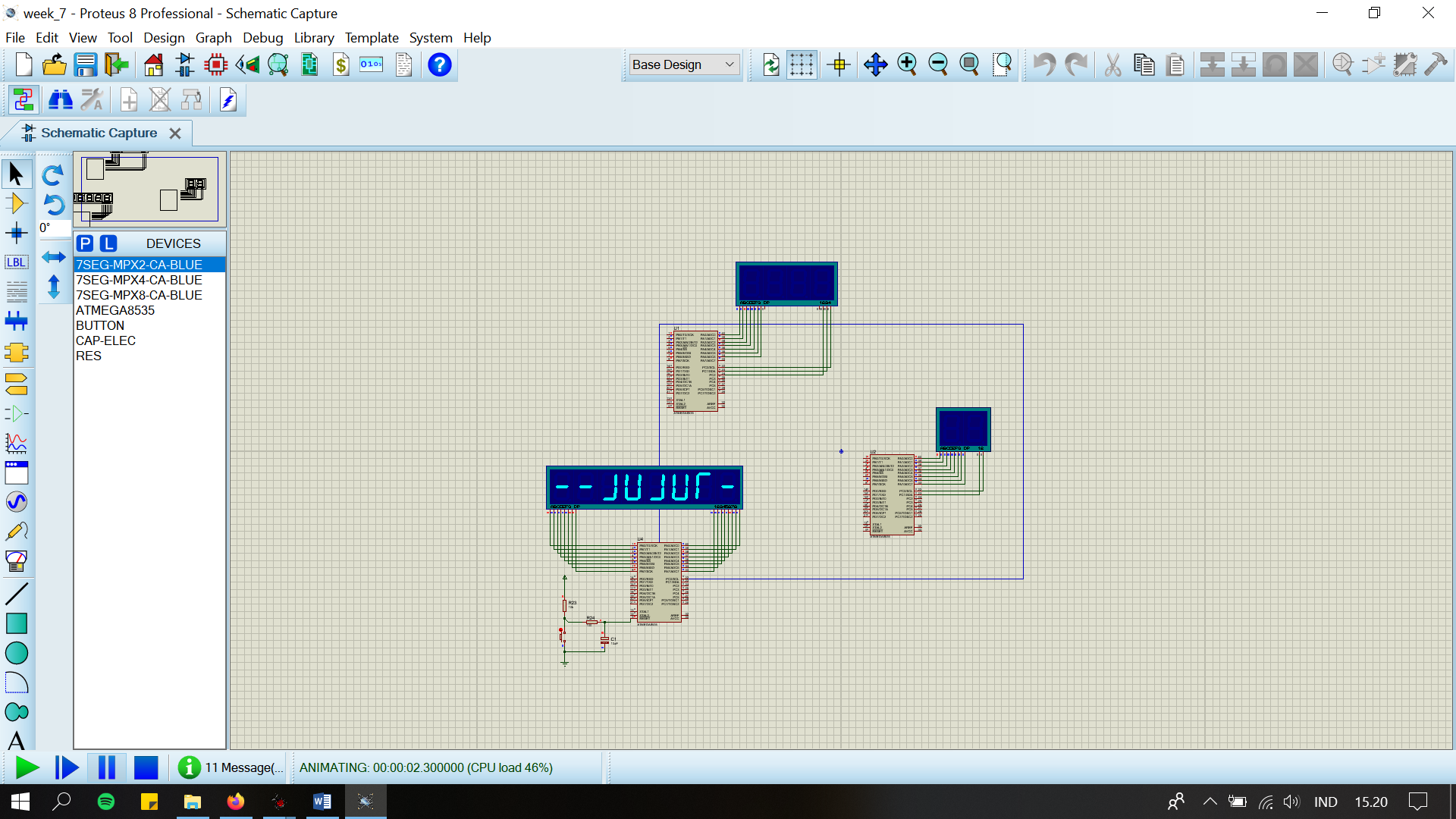
1. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang. Pergantian kata dikendalikan melalui tombol interupsi int0.
2. Buat program untuk menampilkan kata: “--rAJIn-”,”-JUJUr--”, dan “—tULUS-” pada unit peraga 7 segmen (8 karakter) secara bergantian berulang, masing-masing selama 4 detik. Kata “-JUJUr--” berkedip 2 kali tiap detik.

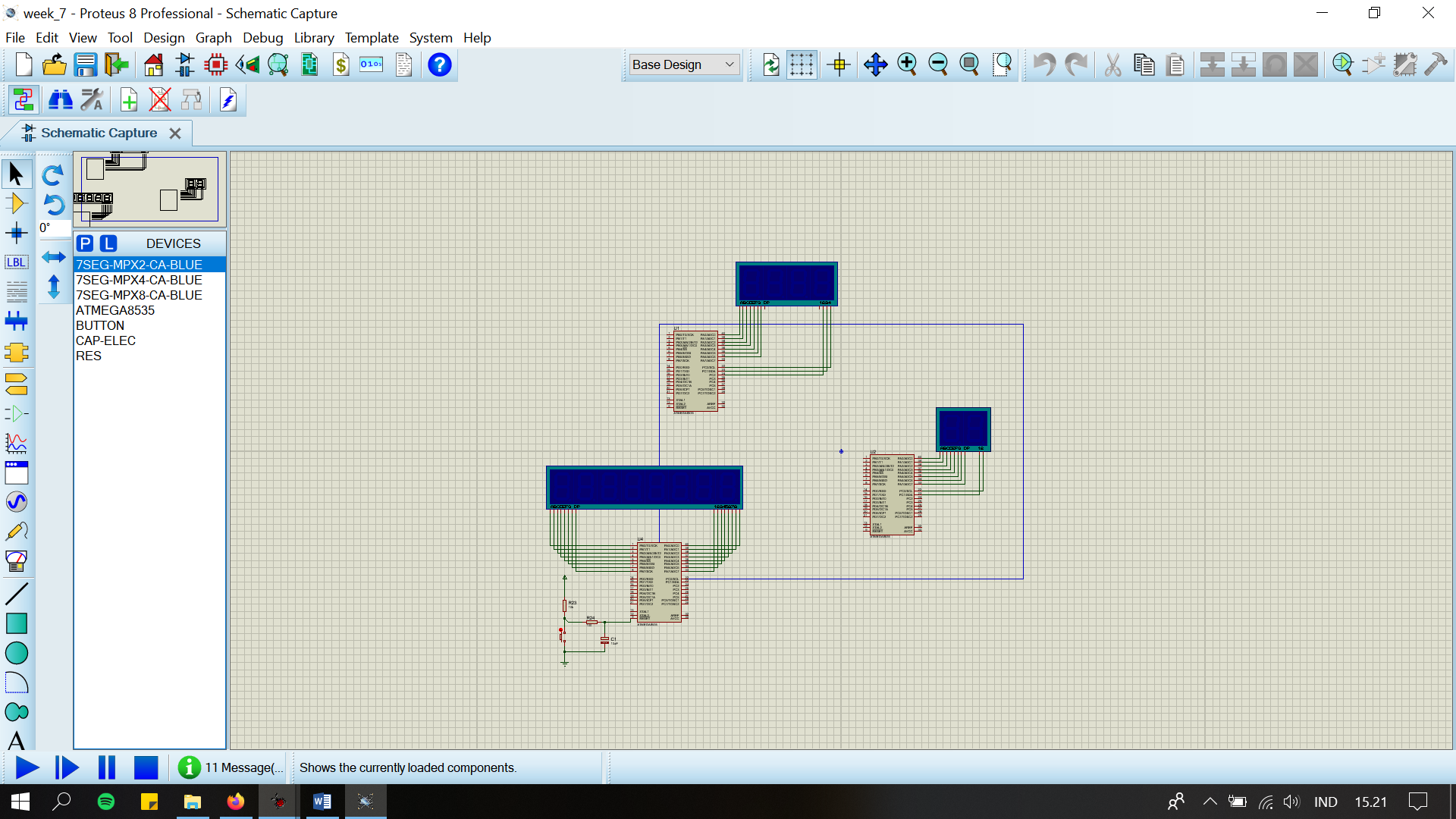
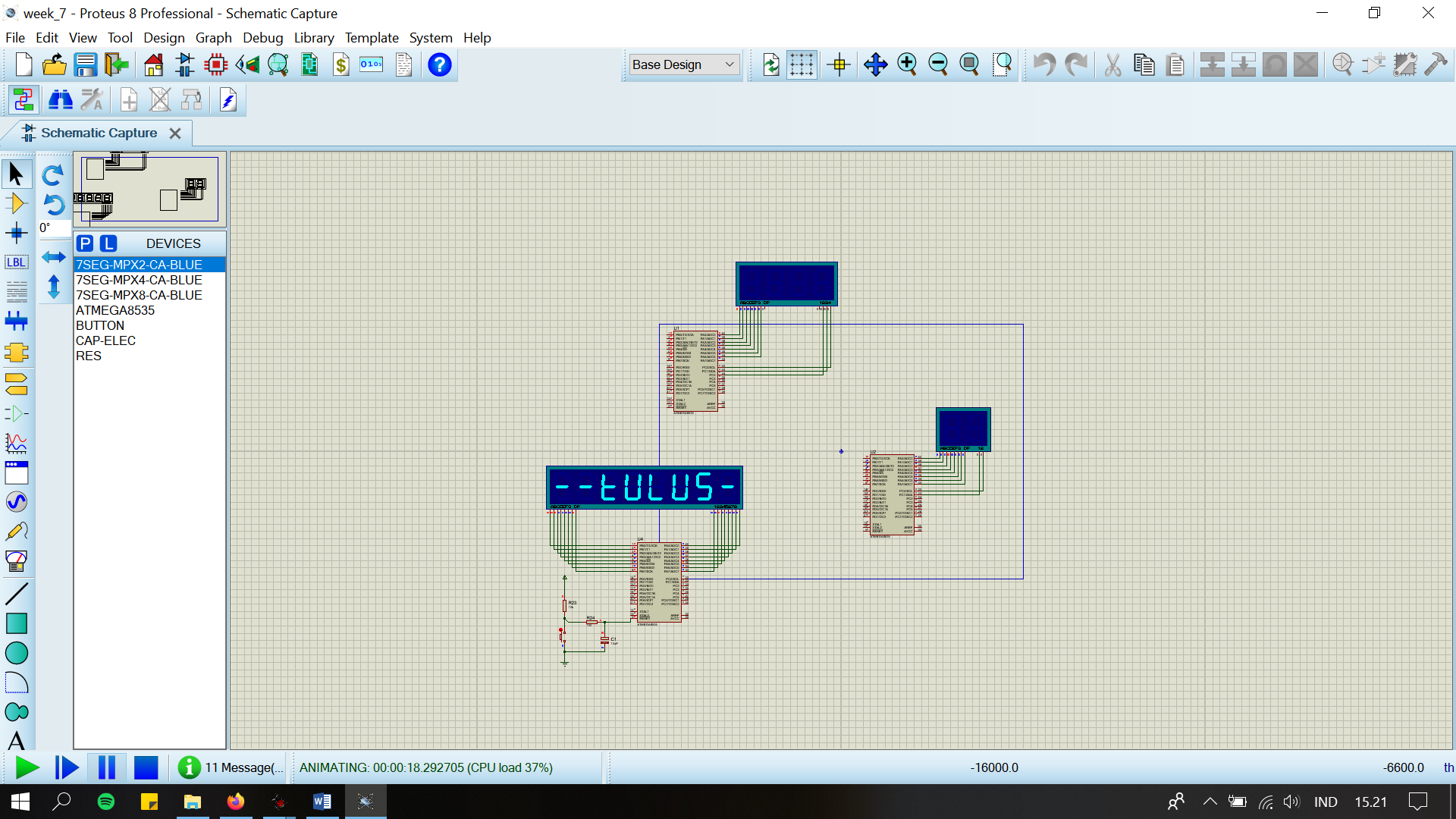
Program:





Hasil:

Hasil Analisis:

Program nomor 12 ini tidak berbeda jauh dengan program no 10 sebelumnya. Perbedaan adalah pada kata “—jujur-“ harus berkedip sekitar 2 kali tiap detik. Pada saat proses kata “—jujur-“ dimulai, dimasukan counter r18 sebanyak 8 kali untuk berkedip 2 kali tiap detik. Counter R16 diisi sebanyak 30, ini berfungsi sebagai counter saat 7 segmentnya mati karena berkedip, agar tidak terlalu lama maka hanya diberi counter sebanyak 30. Diambil kode posisi dan kode karakter kembali sama seperti nomor 10. Pada saat clear display, diberikan delay agar clear display ini tidak terlalu cepat, dilanjutkan dengan pengurangan counter r18 dan memunculkan kata jujur kembali juga dengan clear display nya. Ini membuat kata jujur berkedip. Untuk kata rajin dan tulus bisa disamakan dengan nomor 10.

1. **Kesimpulan**

Kesimpulan pada program ini adalah, untuk memunculkan 1 digit atau lebih penampang 7 segment, bisa digunakan device Multiple Seven Segment baik Common Anode atau Common Cathode. Untuk menggunakannya perlu diperhatikan aturannya, dimana dibutuhkan kode posisi dan kode karakter dalam perancangannya. Perhatikan pula port yang digunakan, dimana dibutuhkan dua port untuk output kode posisi dan kode karakternya.

1. **Daftar Pustaka**

Anonim. 2019. *Decoder TTL BCD ke 7 Segment.* <http://elektronika-dasar.web.id/dekoder-ttl-bcd-ke-7-segment/>. Diakses pada 13 November 2020, pukul 21.10

Djukarna. 2012 . “*AVR Studio 4 dengan bahasa C*” . <https://djukarna.wordpress.com/2012/05/30/avr-studio-4-dengan-bahasa-c/#:~:text=AVR%20studio%20adalah%20sebuah%20software,4%20menggunakan%20bahasa%20pemograman%20Assembler>. Diakses Pada 13 November 2020, pukul 17.35 WIB.