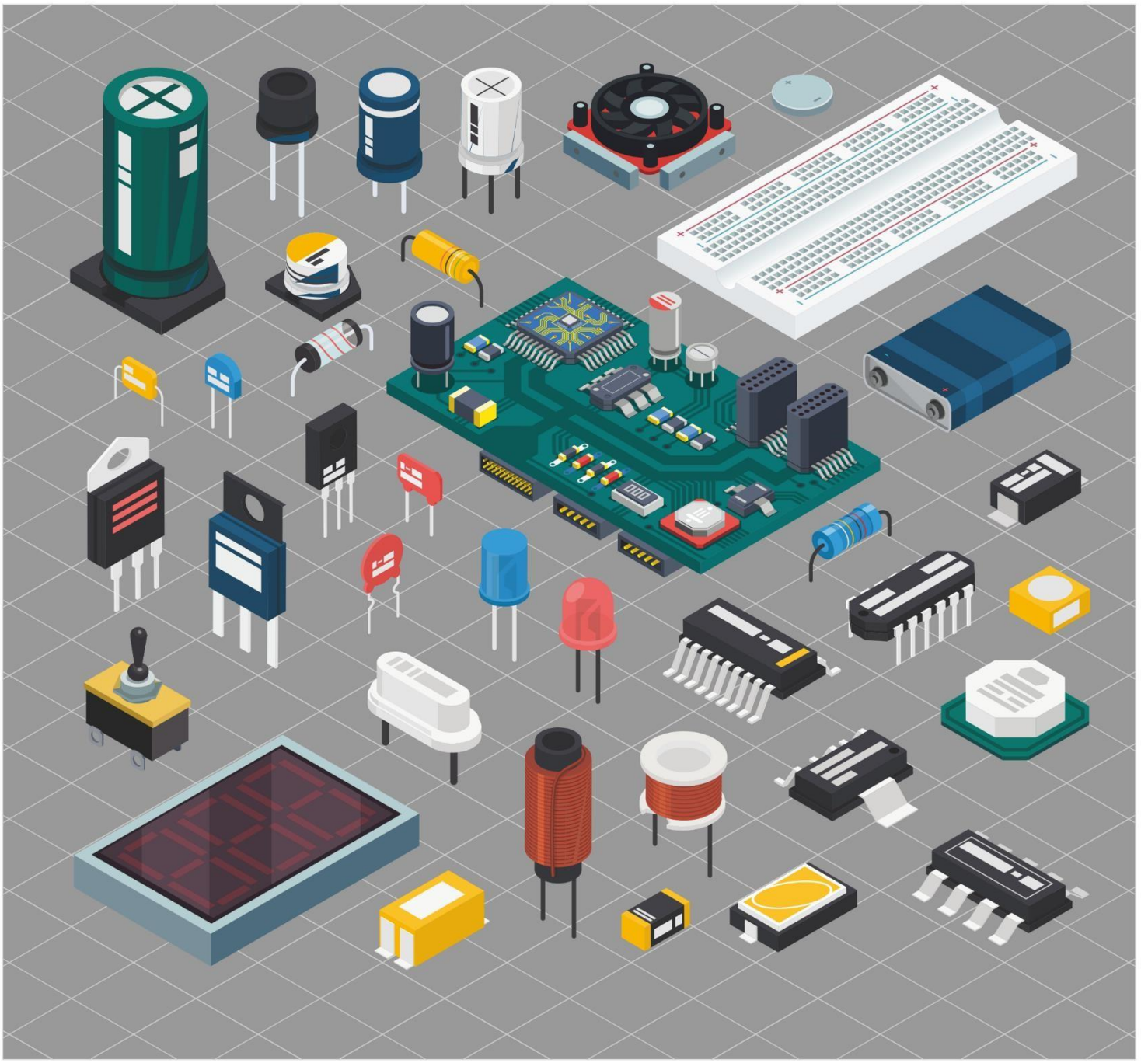

MODUL PRAKTIKUM SISTEM TERTANAM

Robotic and Embedded System Laboratory
Jurusan Teknik Komputer
Universitas Andalas

MODUL 1
ASSEMBLY

1.1 TUJUAN

1. Mengenal dan memahami pemrograman ARM dengan menggunakan bahasa Assemly.
2. Memahami tipe data dan register dalam prosesor ARM.
3. Memahami instruction set dalam processor ARM.
4. Memahami instruksi LDR (Load Register) dan STR (Store Register) dalam memory processor ARM.
5. Memahami dan mengetahui cara Load atau Store Array di assembly ARM

1.2 ALAT DAN BAHAN

1. Laptop
2. Aplikasi PuTTY
3. Raspberry Pi

1.3 LANDASAN TEORI

Assembly

Assembly adalah low-level programming language pada komputer atau pada perangkat lain yang dapat di program. Keunggulan dari Assembly adalah bahasa ini Power Full yang berarti dengan menggunakan program ini, langsung bermain dalam komunikasi antar register. Bahasa Assembly berbeda-beda tergantung arsitektur prosessor dan instruksi set yang digunakan. Agar dapat dieksekusi, program yang ditulis dalam bahasa Assembly harus di compile ke dalam bentuk bahasa mesin. Bahasa Assembly menggunakan mnemonic untuk merepresentasikan low-level machine instruction atau opcode.

Arsitektur ARM

ARM (Advanced RISC Machine) atau arsitektur ARM adalah arsitektur yang mengusung prosesor 32-bit RISC hasil pengembangan dari ARM Limited. Advanced RISC Machine merupakan pengembangan dari Acorn RISC Machine. awalnya, ARM adalah prosesor untuk desktop banyak digunakan oleh keluarga x86. Akan tetapi karena desainnya yang terkesan sederhana, mau tak mau prosesor ini hanya dapat digunakan aplikasi dengan

daya rendah . Karena cocok dengan daya rendah, ARM sering digunakan sebagai prosesor pada pasar mobile electronic serta embedded system. Selain berdaya rendah, kelebihan lainnya terdapat pada harga yang cenderung murah.

ARM banyak diproduksi oleh produsen tingkat dunia. ARM sudah dilisensikan ke berbagai vendor ternama, sebut saja : AMD, NVIDIA, Samsung, serta TI. Karenanya tak heran jika ARM merupakan arsitektur prosesor dengan jumlah produksi terbanyak di dunia.

Raspberry Pi

Raspberry Pi adalah sebuah SBC (Single Board Computer) yang menggunakan processor ARM. ARM merupakan Processor dengan insruction set RISC (Reduced Instuction Set Computing) yang berarti instruksinya kurang dari 100 instruksi. RISC memiliki beberapa kelebihan dan kekurangan. Salah satu kelebihannya adalah instruksi dapat di eksekusi lebih cepat karena sistem RISC memperpendek waktu eksekusi dengan mengurangngi clock cycles. Kekurangannya adalah mengurangi efesiensi penulisan software.

Array

Array merupakan sebuah variable yang menyimpan lebih dari 1 buah data yang memiliki tipe data yang sama. Jadi dapat dikatakan bahwa array merupakan kumpulan dari datadata tunggal yang dijadikan dalam 1 variabel array yang alamat memorinya berbeda yang selanjutnya disebut elemen-elemen array yang bisa kita akses berdasarkan indeks.

Contoh deklarasi array pada bahasa assembly :

```
array_buff:
    .word 0x00000000
    .word 0x00000000
    .word 0x00000000
    .word 0x00000000
```

Tabel 1.1 Register Pada Processor ARM

#	Alias	Purpose
General Purpose		
R0	-	General Purpose
R1	-	General Purpose

R2	-	General Purpose
R3	-	General Purpose
R4	-	General Purpose
R5	-	General Purpose
R6	-	General Purpose
R8	-	General Purpose
R9	-	General Purpose
R10	-	General Purpose
Spesial Purpose		
R7	-	Holds Syscall Number
R11	FP	Frame Pointer
R12	IP	Intra Procedural Pointer
R13	SP	Stack Pointer
R14	LR	Link Pointer
R15	PC	Program Counter
CPSR	-	Current Program Status Registerrm

Tabel 1.2 Instruksi Pada Prosessor ARM

Instruction	Description
MOV	Move Data
MVN	Move 2's complement
ADD	Addition
SUB	Substraction
MUL	Multiplication
LSL	Logical Shift Left
LSR	Logical Shift Right

ASR	Aritmetic Shift Right
ROR	Rorate right
CMP	Compare
AND	Bitwise AND
ORR	Bitwise OR
EOR	Bitwise XOR
LDR	Load
STR	Store
LDM	Load Multiple
STM	Store Multiple
PUSH	Push On Stack
POP	Pop Off Stack
B	Branch
BL	Branch With Link
BX	Branch And Exchange
BLX	Branch With Link And Exchange
SWI/SVC	System Call

1.4 TUGAS PENDAHULUAN

1. Jelaskan apa yang dimaksud dengan SSH!
2. Jelaskan bagaimana Langkah-langkah proses instalasi aplikasi PuTTY berserta screenshot!
3. Jelaskan minimal 10 command pada terminal linux!
4. Jelaskan kelebihan dan kekurangan prosesor ARM!
5. Jelaskan apa pengertian tentang array!
6. Jelaskan bagaimana cara pendeklarasian array dalam assembly!

1.5 PERCOBAAN

A. PROSEDUR PERCOBAAN

1. Hidupkan Laptop
2. Hubungkan dengan jaringan WiFi Labor
3. Buka aplikasi PuTTY dan lakukan komunikasi SSH ke Raspberry Pi dengan memasukkan IP Address : 192.168.0.100, dengan port 22.
4. Login ke Server Raspberry Pi menggunakan akun yang diberikan oleh asisten praktikum
5. Lihat daftar folder menggunakan command:

   ls
6. Masuk ke folder kelompok masing-masing dengan command:

   cd <nama-folder>
7. Ikuti Langkah masing-masing percobaan

B. PERCOBAAN 1 : Register dan Aritmatik Sederhana Dalam Assembly 1

1. Buat sebuat file assembly menggunakan aplikasi nano editor dengan command : **nano <nama_nobp_latihan1>.s**
2. Ketikan program assembly, program diberikan pada saat praktikum.
3. Setelah selesai simpan program menggunakan commans CTRL+S dan keluar dari nano editor menggunakan command CTRL+X.
4. Selanjutnya jalankan command: **as -o <nama_file.o> <nama_file.s>**
5. Lalu ketikan command: **gcc -o <nama_file> <nama-file.s>**
6. Lalu ketikan command **./<nama-file>** untuk mengeksekusi program dan melihat hasilnya serta analisa program diatas.
7. Jalan program pada terminal dan lakukan debugging dengan command:

   gdb ./<nama-file>
8. Di dalam CLI gdb, lakukan perintah:
   a. (gdb) **break *main**
   b. (gdb) **run**
   c. (gdb) **disassemble**
   d. (gdb) **info register**

e. (gdb) **stepi**

9. Ulangi langkah c-e sebanyak line program atau hingga kursor pada command disassamble berada diakhir program.
10. Jalankan command: (gdb) **info register**
11. Lakukan screenshot pada hasil percobaan dan buat analisa jalannya program pada laporan.

C. PERCOBAAN 2 : Operasi Load Sederhana Assembly

1. Buat sebuat file assembly menggunakan aplikasi nano editor dengan command : nano **<nama_nobp_latihan2>.s**
2. Ketikan program assembly, program diberikan pada saat praktikum.
3. Setelah selesai simpan program menggunakan commans CTRL+S dan keluar dari nano editor menggunakan command CTRL+X.
4. Selanjutnya jalankan command: **as -o <nama_file.o> <nama_file.s>**
5. Lalu ketikan command: **gcc -o <nama_file> <nama-file.s>**
6. Lalu ketikan command **./<nama-file>** untuk mengeksekusi program dan melihat hasilnya serta analisa program diatas.
7. Jalan program pada terminal dan lakukan debugging dengan command: **gdb ./<nama-file>**
8. Di dalam CLI gdb, lakukan perintah:
   a. (gdb) **break *main**
   b. (gdb) **run**
   c. (gdb) **disassemble**
   d. (gdb) **info register**
   e. (gdb) **stepi**
9. Ulangi langkah c-e sebanyak line program atau hingga kursor pada command disassamble berada diakhir program.
10. Jalankan command: (gdb) **info register**
11. Lakukan screenshot pada hasil percobaan dan buat analisa jalannya program pada laporan.

D. PERCOBAAN 3 : Operasi Store Sederhana Assembly

1. Buat sebuat file assembly menggunakan aplikasi nano editor dengan command **nano <nama_nobp_latihan3>.s**

2. Ketikan program assembly, program diberikan pada saat praktikum.
3. Setelah selesai simpan program menggunakan commans CTRL+S dan keluar dari nano editor menggunakan command CTRL+X.
4. Selanjutnya jalankan command: **as -o <nama_file.o> <nama_file.s>**
5. Lalu ketikan command: **gcc -o <nama_file> <nama-file.s>**
6. Lalu ketikan command **./<nama-file>** untuk mengeksekusi program dan melihat hasilnya serta analisa program diatas.
7. Jalan program pada terminal dan lakukan debugging dengan command: **gdb ./<nama-file>**
8. Di dalam CLI gdb, lakukan perintah:
   a. (gdb) **break *main**
   b. (gdb) **run**
   c. (gdb) **disassemble**
   d. (gdb) **info register**
   e. (gdb) **stepi**
9. Ulangi langkah c-e sebanyak line program atau hingga kursor pada command disassamble berada diakhir program.
10. Jalankan command: (gdb) **info register**
11. Lakukan screenshot pada hasil percobaan dan buat analisa jalannya program pada laporan.

E. PERCOBAAN 4 : Register dan Aritmatik Sederhana Dalam Assembly 2

1. Buat sebuat file assembly menggunakan aplikasi nano editor dengan command : **nano <nama_nobp_latihan1>.s**
2. Ketikan program assembly, program diberikan pada saat praktikum.
3. Setelah selesai simpan program menggunakan commans CTRL+S dan keluar dari nano editor menggunakan command CTRL+X.
4. Selanjutnya jalankan command: **as -o <nama_file.o> <nama_file.s>**
5. Lalu ketikan command: **gcc -o <nama_file> <nama-file.s>**
6. Lalu ketikan command **./<nama-file>** untuk mengeksekusi program dan melihat hasilnya serta analisa program diatas.

7. Jalan program pada terminal dan lakukan debugging dengan command: **gdb ./<nama-file>**
8. Di dalam CLI gdb, lakukan perintah:
   a. (gdb) **break *main**
   b. (gdb) **run**
   c. (gdb) **disassemble**
   d. (gdb) **info register**
   e. (gdb) **stepi**
9. Ulangi langkah c-e sebanyak line program atau hingga kursor pada command disassamble berada diakhir program.
10. Jalankan command: (gdb) **info register**
11. Lakukan screenshot pada hasil percobaan dan buat analisa jalannya program pada laporan.